DOCUMENT 128-17

Telemetry Group

INSTRUMENTATION HARDWARE ABSTRACTION LANGUAGE (IHAL) HANDBOOK

ABERDEEN TEST CENTER
DUGWAY PROVING GROUND
REAGAN TEST SITE
REDSTONE TEST CENTER
WHITE SANDS MISSILE RANGE
YUMA PROVING GROUND

NAVAL AIR WARFARE CENTER AIRCRAFT DIVISION
NAVAL AIR WARFARE CENTER WEAPONS DIVISION
NAVAL UNDERSEA WARFARE CENTER DIVISION, KEYPORT
NAVAL UNDERSEA WARFARE CENTER DIVISION, NEWPORT
PACIFIC MISSILE RANGE FACILITY

30TH SPACE WING
45TH SPACE WING
96TH TEST WING
412TH TEST WING
ARNOLD ENGINEERING DEVELOPMENT COMPLEX

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

This page intentionally left blank

DOCUMENT 128-17

INSTRUMENTATION HARDWARE ABSTRACTION LANGUAGE (IHAL) HANDBOOK

January 2017

Prepared by

Telemetry Group

Published by

Secretariat
Range Commanders Council
White Sands Missile Range
New Mexico 88002-5100

This page intentionally left blank.

Table of Contents

Preface ..... **vii**

Acronyms ..... **ix**

Chapter 1. Introduction **1-1**
1.1 The Range Commanders Council (RCC) and IHAL 1-1
1.2 Problem Description 1-2
1.3 IHAL Use Cases 1-5
1.3.1 IHAL as a Description Language 1-5
1.3.2 IHAL as a Command Language 1-5
1.4 eXtensible Markup Language (XML) 1-6

Chapter 2. IHAL Basics **2-1**
2.1 IHAL XML Key Concepts 2-1
2.2 Generic Constructs in IHAL 2-2
2.3 Utilization of Third-Party Standards 2-2
2.4 IHAL Schema Overview 2-2
2.4.1 Attributes 2-4
2.4.2 Generic Instruments 2-6
2.4.3 Generic Functions 2-9
2.4.4 Generic Channels 2-10

Chapter 3. Getting Started with a Simple Example **3-1**
3.1 Basic Instrumentation Graph 3-1
3.2 Data Acquisition Unit 3-2
3.3 Signal Conditioning 3-6
3.4 Temperature Measurement 3-12

Chapter 4. The IHAL API **4-1**
4.1 Errors 4-3
4.2 API Functions 4-3
4.2.1 Retrieve a Vendor's Pool 4-3
4.2.2 Retrieve the List of Available Configurations 4-3
4.2.3 Retrieve a Specific Configuration 4-4
4.2.4 Change the Value of a Configurable Attribute 4-4
4.2.5 Create a New Configuration 4-4
4.2.6 Add a Device to a Configuration 4-5
4.2.7 Remove a Device from a Configuration 4-5
4.2.8 Program the Hardware 4-5
4.2.9 Add a New format to a Data Encoder 4-5
4.2.10 Add a Measurement to an Existing Format 4-6
4.2.11 Remove a Measurement from a Format 4-6

Appendix A. Citations A-1

Table of Figures

Figure 1-1. Motivation for the Development of IHAL 1-3
Figure 1-2. Neutral Language (IHAL) as a Solution to the m*n Problem 1-4
Figure 1-3. Use of IHAL with a Configuration API 1-5
Figure 1-4. XML Snippet 1-6
Figure 1-5. Components of an XML Element 1-7
Figure 1-6. Example Schema Diagram 1-8
Figure 2-1. Logical Model of an IHAL Device 2-2
Figure 2-2. Top-Level IHAL Schema Diagram 2-3
Figure 2-3. Numeric Attribute Schema Diagram 2-4
Figure 2-4. Possibly Configurable Attribute Type Schema Diagram 2-5
Figure 2-5. Custom Attribute Schema Diagram 2-6
Figure 2-6. Generic IHAL Device Schema Diagram 2-7
Figure 2-7. Device Description Schema Diagram 2-7
Figure 2-8. Dimensions Schema Diagram 2-7
Figure 2-9. Connector Schema Diagram 2-7
Figure 2-10. Environmental Characteristics Schema Diagram 2-8
Figure 2-11. Generic Instrument Schema Diagram 2-9
Figure 2-12. Generic Function Schema Diagram 2-10
Figure 2-13. Generic Channel Schema Diagram 2-11
Figure 3-1. A Simple IHAL Example XML Schema 3-1
Figure 3-2. A Simple IHAL Example 3-2
Figure 3-3. Partial DAU Schema 3-3
Figure 3-4. Instrument Pool with a single DAU. 3-4
Figure 3-5. Instrument Pool DAU Function Characteristics 3-4
Figure 3-6. Instrument Use Set Attribute Schema 3-5
Figure 3-7. Example Instrumentation Graph with DAU Added 3-6
Figure 3-8. Analog Signal Conditioning Card XML Schema 3-6
Figure 3-9. Analog Signal Conditioning Function XML Schema 3-7
Figure 3-10. Analog Signal Conditioning Function Attributes XML Schema 3-7
Figure 3-11. Analog Signal Conditioning Function Sub-Functions XML Schema 3-7
Figure 3-12. XML Snippet with Analog Signal Conditioning Card Descriptive Information 3-8
Figure 3-13. XML Snippet with Analog Signal Conditioning Card Function Information 3-8
Figure 3-14. XML Snippet with Analog Signal Conditioning Card Sub-Function Information 3-9
Figure 3-15. Channel Use Set Attribute Schema 3-10
Figure 3-16. Analog signal conditioning card added to the instrumentation graph. 3-11
Figure 3-17. Connection Schema 3-12
Figure 3-18. Connecting the Card to the DAU 3-12
Figure 3-19. Thermocouple XML Schema 3-12
Figure 3-20. Measurement Channel XML Schema 3-13

Figure 3-21. Measurement Function Characteristics XML Schema 3-14
Figure 3-22. Thermocouple Channel XML Schema 3-14
Figure 3-23. XML Snippet with Thermocouple Descriptive Information 3-15
Figure 3-24. XML Snippet with Thermocouple Function Information..... 3-15
Figure 4-1. Envisioned Use Cases for the IHAL API 4-2

List of Tables

Table 3-1. Configuration/Instrumentation Graph Datasheet Information..... 3-1
Table 3-2. DAU Datasheet Information..... 3-3
Table 3-3. DAU Use Information..... 3-5
Table 3-4. Signal Conditioning Card Datasheet Information..... 3-7
Table 3-5. Signal Conditioning Card Channel Use Information..... 3-9
Table 3-6. Thermocouple Datasheet Information 3-13

This page intentionally left blank.

Preface

This Instrumentation Hardware Abstraction Language (IHAL) Handbook was prepared by the Data Multiplex Committee of the Telemetry Group (TG), Range Commanders Council (RCC). The IHAL Handbook is a common instrumentation reference for use by organizations that produce IHAL files, by ranges that receive IHAL files, and by vendors who incorporate IHAL files into their telemetry processing systems. The use of this handbook will eliminate inconsistencies and differing interpretations of IHAL files so that all parties will benefit from its usage.

The RCC gives special acknowledgement for production of this document to the TG Data Multiplex Committee. Please direct any questions to the committee point of contact or to the RCC Secretariat as shown below.

Telemetry Group Data Multiplex Committee Chairman: Mr. Jon Morgan
412 TW, Edwards AFB
Bldg 1408 Room 5
301 East Yeager
Edwards AFB, CA 93524
Phone: DSN 527-8942 Com (661) 277-8942
Fax: DSN 527-8933 Com (661) 277 8933
email jon.morgan.2.ctr@us.af.mil

Secretariat, Range Commanders Council
ATTN: CSTE-WS-RCC
1510 Headquarters Avenue
White Sands Missile Range, New Mexico 88002-5110
Phone: DSN 258-1107 Com (575) 678-1107
Fax: DSN 258-7519 Com (575) 678-7519
email usarmy.wsmr.atec.list.rcc@mail.mil

This page intentionally left blank.

Acronyms

API	application programming interface
DAU	data acquisition unit
DM	data multiplex
HTTP	Hypertext Transfer Protocol
IHAL	Instrumentation Hardware Abstract Language
iNET	integrated Network Enhanced Telemetry
IRIG	Inter-Range Instrumentation Group
ISS	Instrumentation Support System
ITC	International Telemetering Conference
MDL	Metadata Description Language
PCM	pulse code modulation
RCC	Range Commanders Council
REST	representational state transfer
RPC	remote procedure call
SOAP	Simple Object Access Protocol
T&E	test and evaluation
TG	Telemetry Group
TMATS	Telemetry Attributes Transfer Standard
URL	Uniform Resource Locator
VIT	Vehicular Instrumentation/Transfer
W3C	World Wide Web Consortium
XidML	eXtensible Instrumentation Definition Markup Language
XML	eXtensible Markup Language

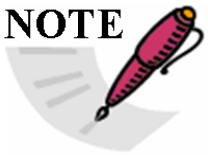
This page intentionally left blank.

CHAPTER 1

Introduction

This Instrumentation Hardware Abstraction Language (IHAL) Handbook is intended to supplement Chapter 9[1] of RCC IRIG 106 Telemetry Standards. Practical guidance for properly generating and using IHAL files is provided and examples of some of the more commonly used IHAL features are given. Since there may be multiple ways of describing these features in IHAL, the examples are intended to illustrate "best practices." The overall purpose of this handbook is to improve the use of IHAL as a standard by presenting clear guidelines and thereby eliminating any misinterpretations that may exist.

The RCC IRIG 106 sets forth standards for various aspects of telemetry (TM) data transmission, recording, and processing. These standards constitute a guide for the orderly implementation of TM systems and provide the necessary criteria on which to base equipment design and modification. Their purpose is to ensure efficient spectrum utilization, interference-free operation, interoperability between ranges, and compatibility of range user equipment at the ranges.

NOTE 	The RCC IRIG 106 is the master source of all information for TM data transmission, recording, and processing. Therefore, the RCC IRIG 106 is assumed to be correct if a discrepancy is found between it and this handbook. If a discrepancy is found, it should be immediately reported to the RCC Secretariat or to the Telemetry Group (TG). The RCC IRIG 106 can be viewed or downloaded from the RCC public website, http://www.wsmr.army.mil/RCCsite/Pages/default.aspx.

1.1 The Range Commanders Council (RCC) and IHAL

The RCC held its first meeting in August 1951. In March 1952, the RCC Commanders established the Inter-Range Instrumentation Group (IRIG) to make recommendations for improvement of range instrumentation and conservation of the resources of the ranges. After a few meetings, the IRIG recognized the need to expand and specialize, and the IRIG Steering Committee was created to oversee several IRIG technical working groups. In 1971, the IRIG Steering Committee was disbanded, and the IRIG working groups became known as the RCC working groups. To this day, the RCC standards documents are still commonly referred to as IRIG standards.

In October 2010, the Vehicular Instrumentation/Transducer (VIT) Committee of the RCC TG initiated a task to develop an XML format for describing instrumentation hardware. The initial requirement was for the task to satisfy the original intention of the Telemetry Attributes Transfer Standard (TMATS) 'H' group while making use of the previous work done in developing IHAL. The TMATS 'H' group has been added as a placeholder for hardware descriptions, but has never been implemented.

[1] Range Commanders Council. "Telemetry Attributes Transfer Standard," in *Telemetry Standards*. IRIG 106-15. July 2015. May be superseded by update. Retrieved 1 July 2015. Available at http://www.wsmr.army.mil/RCCsite/Documents/106-15_Telemetry_Standards/Chapter9.pdf

The VIT Committee formed a working group consisting of committee members, original IHAL developers, instrumentation vendors, and other government and commercial members of the test and evaluation (T&E) community. At the time the task was initiated, IHAL had been shown to support configuration of analog signal conditioning hardware and pulse code modulation (PCM) data encoders. The working group first reviewed the existing IHAL design in detail, making some minor changes to the fundamental structure of the language and to the way analog signal conditioners are described. The group then expanded the language with support for additional hardware types, including bus monitor cards, data acquisition units (DAUs), network data encoders, and recorders.

Task work was carried out during the RCC TG's twice-a-year meetings and also during monthly teleconferences with the working group. A web-based portal was maintained for sharing the latest schema, documentation, and meeting minutes. The portal also included discussion forums, enabling working group members to continue discussing the emerging standard between meetings.

Following the first year of standardization work, the IHAL language and application programming interface (API) were validated through a multi-vendor demonstration held at the 2011 International Telemetering Conference (ITC). To support this demo, two different instrumentation hardware vendors implemented the IHAL API for a subset of their hardware. In the demonstration scenario, each vendor configuration tool was pre-loaded with an IHAL configuration consisting of one DAU connected to three different analog signal conditioning cards, for a total of seven unique devices from two different vendors. Each vendor configuration tool was queried for the current IHAL configuration, and both IHAL configurations were displayed in a single view. The settings on each device were then changed and immediately communicated to the appropriate vendor configuration tool through the IHAL API. The vendor configuration tools in turn validated the change and provided immediate feedback about the impact to other settings.

The IHAL standard consists of both an XML-based language and an API specification, each of which are illustrated below. The IHAL standard was published in 2013 as Section 9.7 of Chapter 9 (Range Commanders Council 2015).

1.2 Problem Description

Current instrumentation support systems (ISSs) use vendor-specific languages in order to support the programming of instrumentation systems prior to instrumentation testing. Since there is a number of vendors selling various data acquisition, control, transmission, and storage components, an ISS would not only have to interface with these different systems but also with the different vendors that sell various instrumentation systems. Since it is practically impossible for a single ISS to interface with every known system and vendor, the ISSs typically select only a few and offer an almost complete solution for those hardware vendors. This hinders the ability to mix and match vendors and develop a robust and cost-effective solution.

A second issue involves format change. Any hardware vendor format change or functional enhancement requires a change in the ISS. Even if these change considerations were to be deemed financially feasible by the ISS vendors, they would be incorporated into the next release after a considerable period of time.

The description of the problem is illustrated in Figure 1-1.

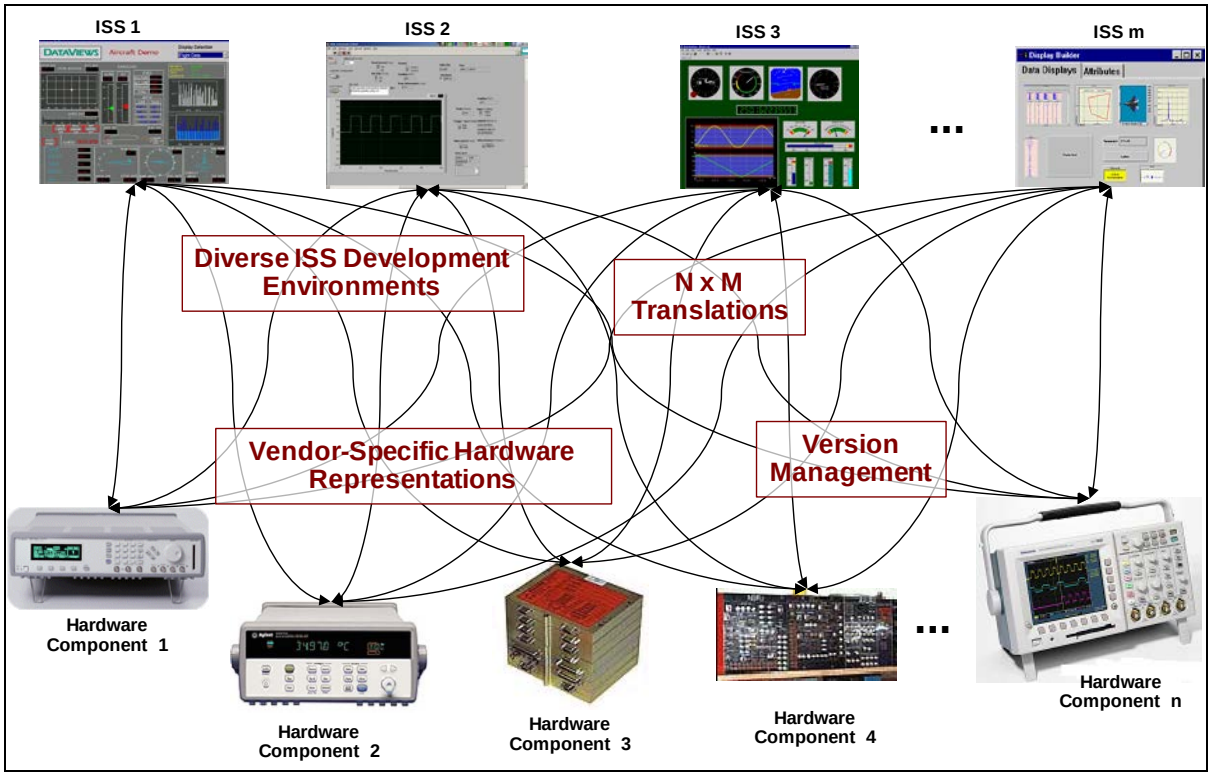


Figure 1-1. Motivation for the Development of IHAL

These issues can be mitigated by developing a neutral or common language that acts as an intermediary between the ISS and each hardware instrument vendor format. This neutral language should abstract various hardware characteristics and be a "least common multiple" among all hardware types that form an instrumentation system. In that sense, the neutral language acts as the intermediate language between various high-level languages and various vendor-specific machine languages. Using such a language, it is possible to reduce the N*M translations as shown in Figure 1-1 to N+M translations, where M is the number of ISSs and N is the number of instruments.

The IHAL vision is that a single language can be used to specify instrumentation network and hardware configurations as well as act as a command-and-query language. This provides a more elegant and robust solution for specifying, simulating, managing, querying, and controlling T&E instrumentation networks and systems. The use of IHAL is shown in Figure 1-2.

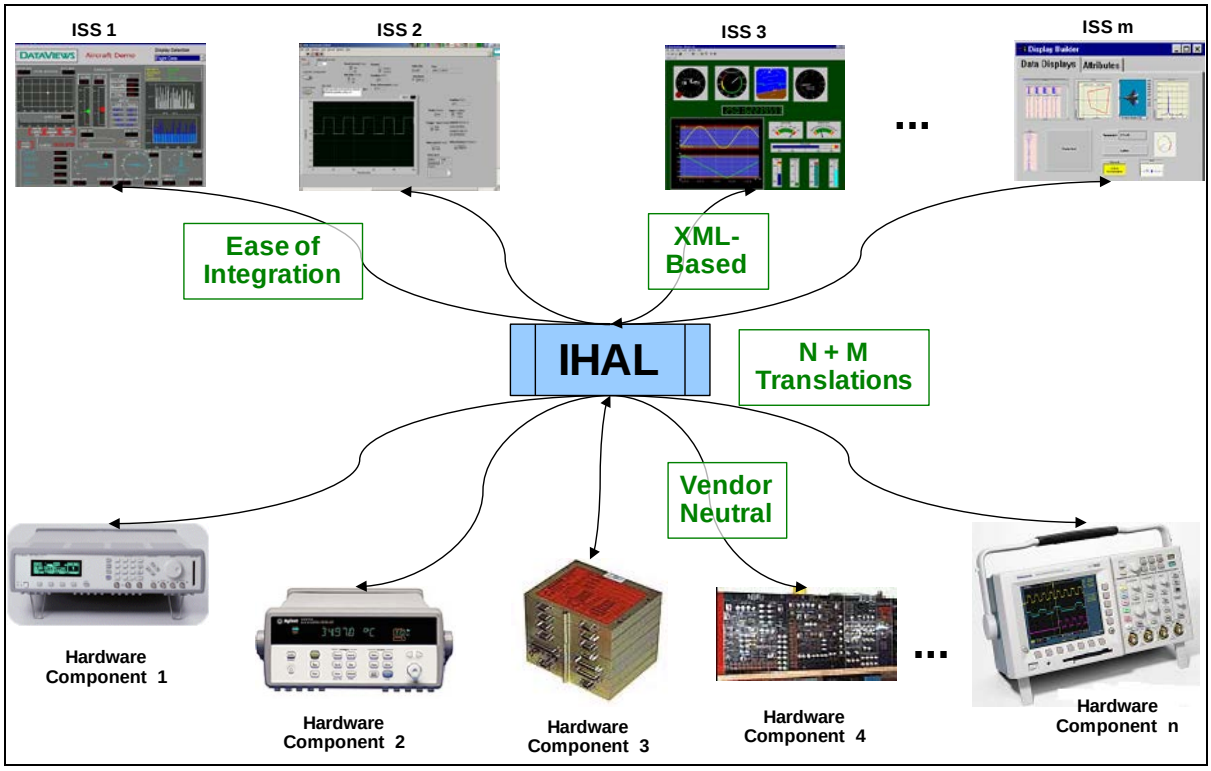


Figure 1-2. Neutral Language (IHAL) as a Solution to the m*n Problem

By utilizing a standard language such as IHAL, an ISS must understand only a single representation in order to support all IHAL-capable hardware vendors. This enables instrumentation engineers to be trained on only one IHAL-aware application and be able to configure multi-vendor instrumentation systems with little or no additional training

At a larger scale, such a language can be used in diverse TM, instrumentation, and remote sensing areas such as process control, sensor networks relating to homeland security, and mission-critical applications in space, aeronautics, healthcare, and manufacturing.

Additionally, developing a configuration API that works in conjunction with the IHAL standard allows third-party software to interact directly with the vendor's configuration software without requiring usage of the vendor's software user interface; this vision is shown in Figure 1-3.

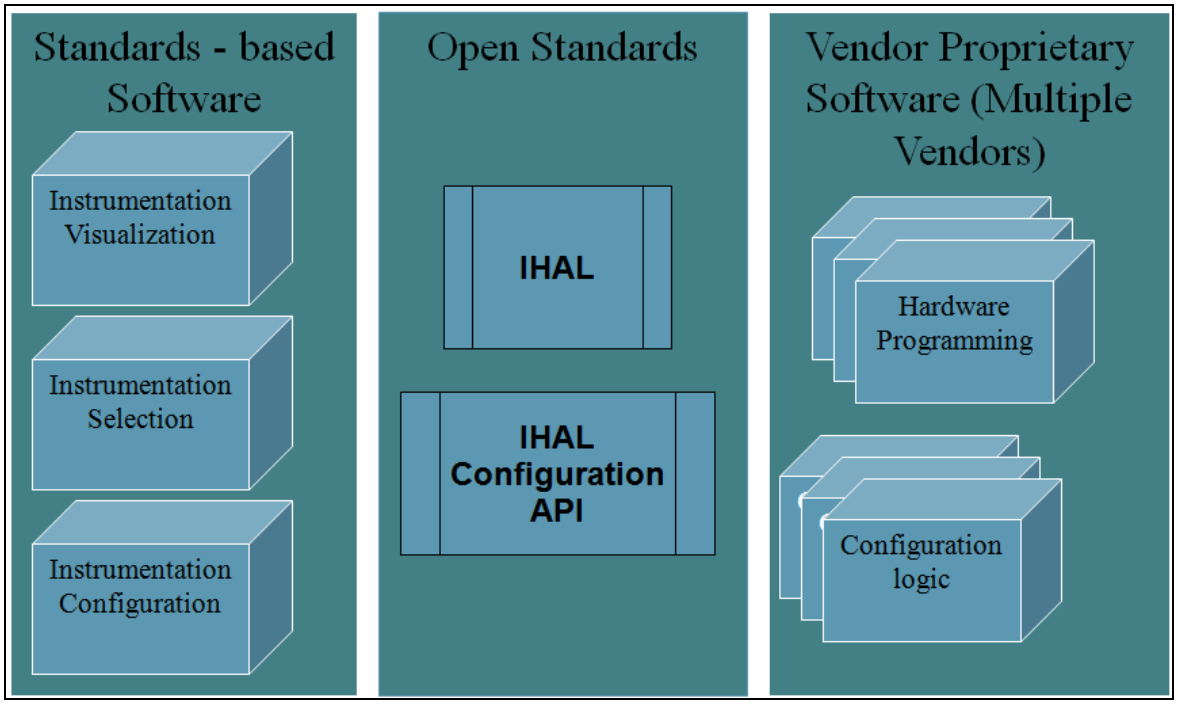


Figure 1-3. Use of IHAL with a Configuration API

1.3 IHAL Use Cases

The potential uses of IHAL fall into two major categories: 1) IHAL as a description language; and 2) IHAL as a command language.

1.3.1 IHAL as a Description Language

As a vendor-neutral, human-readable language for describing instrumentation hardware, IHAL provides a means for storing a permanent record of the devices and their settings during a test. This description will remain readable and relevant even if the vendors who made the hardware cease to exist or radically change their file formats.

Additionally, providing such descriptions enables the development of vendor-neutral tools. The capabilities of these tools can range anywhere from simple visualization (e.g., instrumentation network and configuration visualization) to complex automated reasoning (e.g., automatically select and configure devices from multiple vendors based on some user-defined requirements).

1.3.2 IHAL as a Command Language

The same constructs in IHAL that describe the current configuration of a device can also be used to issue a "command" to the device to change its configuration. When combined with the API (described above), this feature of IHAL enables multi-vendor instrumentation configuration from a single user interface, without requiring vendors to share knowledge about the internal workings of their configuration engines.

1.4 eXtensible Markup Language (XML)

The XML specification is produced by the World Wide Web Consortium (W3C)[2] whose original intent was to provide a standard, machine-readable format for describing documents. Because of its popularity, wide adoption, and prevalence on the Internet, its use has expanded to describe arbitrary data structures such as web services and T&E metadata. Here we provide a brief overview of XML. More detail can be found in the RCC Style Guide.[3]

The example in Figure 1-4 shows a portion of an XML document (an XML snippet).

```
<ihalinstgraph:instrumentationGraph ihalcommon:ID="graph1">
  <ihalinstgraph:description>
    <ihalcommon:name>
      Instrumentation Graph for Handbook Example
    </ihalcommon:name>
  </ihalinstgraph:description>
  <ihalinstuse:instrumentUse ihalcommon:ID="dauUse1" ihalcommon:Ref="dau1">
    <ihalinstuse:serialNumber>A0000001</ihalinstuse:serialNumber>
    <ihalinstuse:location>Behind cockpit</ihalinstuse:location>
    <ihalinstuse:attributeSettings>
      <ihalinstuse:setAttribute
         ihalcommon:ID="setMasterSlaveMode1"
         ihalcommon:Ref="dau1-masterSlaveMode">
        <ihalinstuse:setConfigurableEnumeratedAttribute>
          <ihalinstuse:stringValue>Standalone</ihalinstuse:stringValue>
        </ihalinstuse:setConfigurableEnumeratedAttribute>
      </ihalinstuse:setAttribute>
    </ihalinstuse:attributeSettings>
  </ihalinstuse:instrumentUse>
</ihalinstgraph:instrumentationGraph>
```

Figure 1-4. XML Snippet

In XML, each piece of data, or element, is surrounded by a "tag" such as `<ihalinstgraph:instrumentationGraph>` and `<ihalinstuse:setAttribute>`. The structure of an XML file is such that tags can be enclosed in other tags to an arbitrary depth (`<ihalcommon:name>` is a sub-element of `<ihalinstgraph:description>`, etc.). This is the basic idea behind the structure of an XML document.

The component parts of an XML element are identified in Figure 1-5. Each of these components is defined below the figure.

[2] http://www.w3.org/

[3] Range Commanders Council. *XML Style Guide*. RCC 125-15. July 2015. Retrieved 13 January 2017. Available at http://www.wsmr.army.mil/RCCsite/Documents/125-15_XML_Style_Guide/.

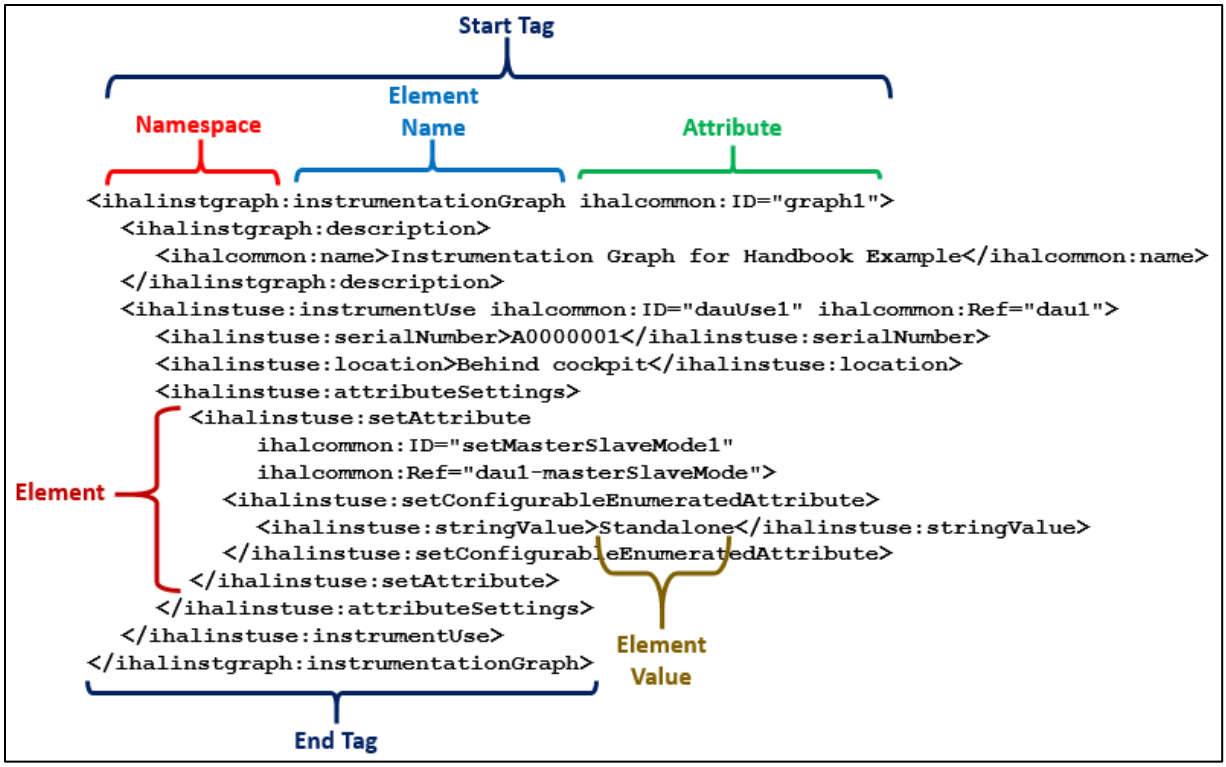


Figure 1-5. Components of an XML Element

- **Element:** "Element" is the term used to define a complete unit of XML information. It begins with a start tag and ends with an end tag. The value of an element can be a simple value or one or more sub-elements (children).
- **Start Tag:** The start tag identifies the beginning of the element and consists of the element's name (and possibly a namespace and attributes) included between a "<" and a ">" symbol.
- **End Tag:** The end tag identifies the end of the element and looks identical to the start tag, except it includes a "/" (forward-slash) after the "<" symbol. An end tag does not contain attributes.
- **Namespace:** The namespace is optional in XML, but can be used to define the scope within which the element is defined. In our IHAL example, we define a "ihalinstuse" namespace (for use information) and make all of the elements for describing instrument usage members of it.
- **Element name:** The name of the element is what appears in the start and end tags and is what actually identifies the piece of information being defined.
- **Attribute:** Attributes are another method of associating information with an XML element. An attribute consists of a name followed by a "=" sign followed by a value enclosed in quotes.
- **Element value:** The value of the element is everything that lies between the start tag and the end tag. The value can either be a single value (e.g. 7, "John", true, etc.) or a collection of one or more sub-elements (children).

An XML schema is a design document used to describe a specific language that is based on XML. The rules for formatting proper XML are very simple and unrestricted. A schema defines which element names are valid, which elements can have which children, and which values are valid for each element. Types organize XML documents by defining the allowed structure for specific groupings of elements.

Even though an XML schema is itself a document, it is usually more useful to view the schema as a diagram. In this representation, boxes represent XML schema complex types (types that have structure). The bold title in the box is the name of the type. Inside the box, attributes are shown under the *Attributes* heading and simple types (types that do not have structure) are shown under the *Elements* heading. Elements that are complex types are represented by a line that connects the element with its type. The line is labeled with the name of the element. Figure 1-6 contains an example schema diagram.

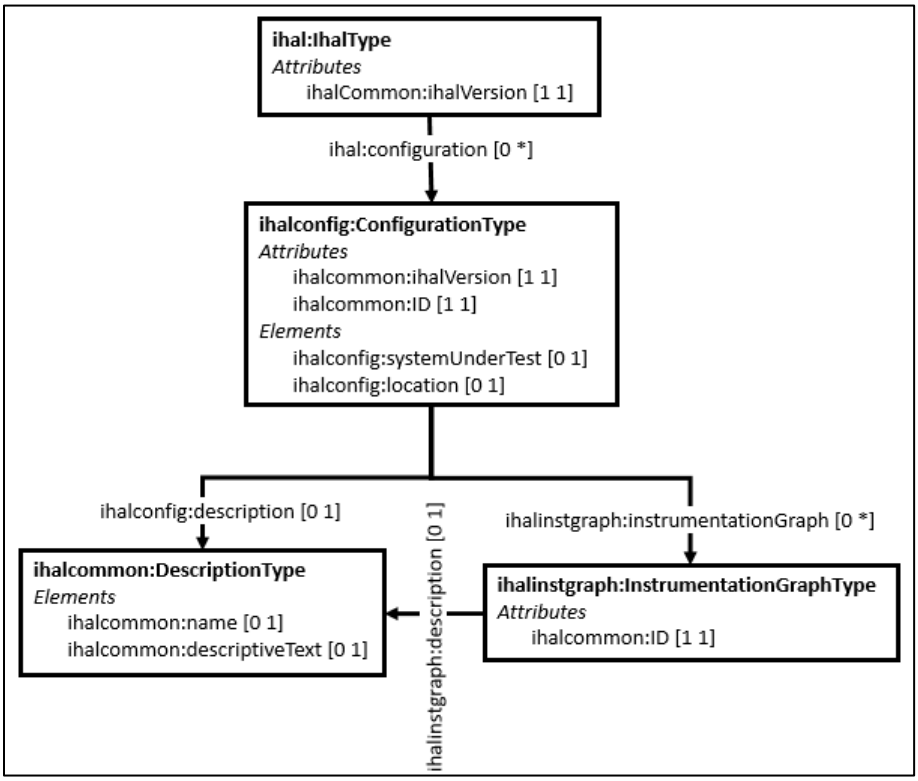


Figure 1-6. Example Schema Diagram

CHAPTER 2

IHAL Basics

In this section we describe the key concepts in the IHAL schema. In the next section, we illustrate the usage of the schema through simple IHAL examples.

2.1 IHAL XML Key Concepts

The IHAL language is focused on the settings available on devices, referred to as "configurable attributes." The primary purpose of IHAL is to describe these attributes both in terms of how they are currently configured and how they can be configured.

One important concept in IHAL is that of "pool"- versus "use"-level device descriptions. The IHAL instrument pool is used to describe instrumentation hardware that is available for use in a configuration. At the pool level, each device is described in detail with all of the information you would typically find in a vendor's spec (or cut) sheet, such as the settings, size, weight, environmental characteristics, connectors, etc. Pool-level descriptions are independent of any specific configuration, and describe what the device does (i.e., what functions it performs) and what settings are available, as well as what the valid values are for each setting.

On the other hand, use-level descriptions define a specific instance of a device along with the current values for each setting in a specific configuration. One way to understand the difference between pool- and use-level descriptions is to think of pool-level devices as being uniquely identified by a model number and use-level devices by a serial number.

Another key concept is the hardware function, which in IHAL represents a specific capability provided by a device. Each function can contain attributes (configurable and non-configurable) as well as sub-functions. For example, the analog signal conditioning function can have configurable attributes such as minimum and maximum input voltages and an offset. The analog signal conditioning function can also contain amplification, filtering, and analog-to-digital conversion sub-functions.

A device in IHAL may also be composed of one or more hardware channels, which in IHAL define a collection of functionality that may be repeated multiple times within the device. Channels in IHAL are directly analogous to the channels on a card. Channels in IHAL are always composed of functions, but not all functions are part of a channel. Some functions may be associated directly with the device itself.

Another way in which functionality can be grouped in IHAL is through a format, which is used only in the description of data encoders and are used to describe the configurability of a data format. For example, a PCM data encoder can contain one or more PCM formats, which in turn are described in terms of the PCM encoder function and associated attributes such as number of minor frames, word length, and parity.

The complete logical model of instrumentation devices in IHAL is illustrated in Figure 2-1. This shows how a device in IHAL can be decomposed into functions, channels, and formats.

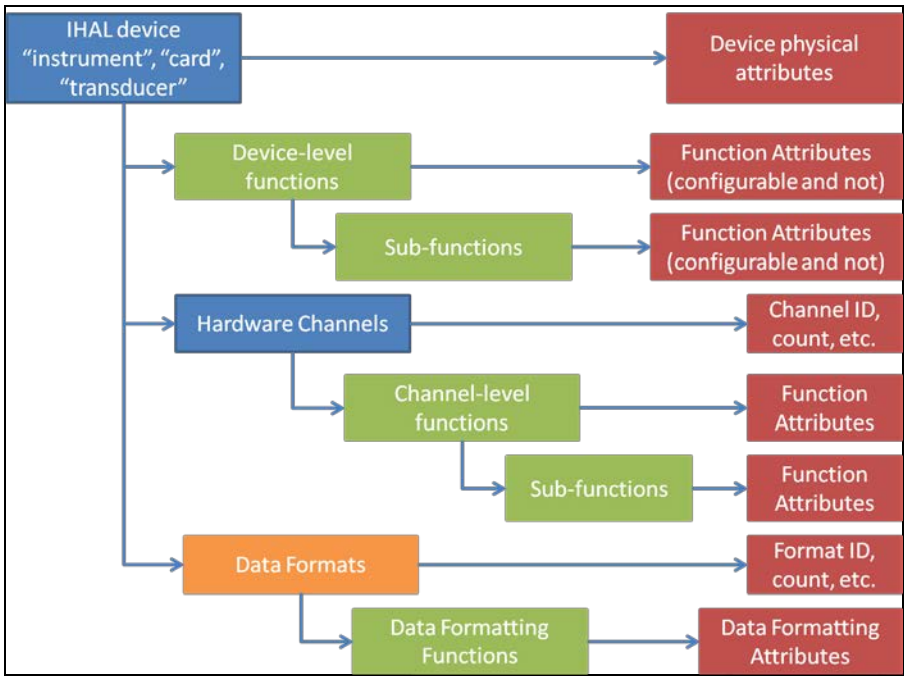


Figure 2-1. Logical Model of an IHAL Device

2.2 Generic Constructs in IHAL

Much care has gone into the design of the IHAL schema to ensure that there is a generic structure defined for each key concept, and that more specific concepts can "inherit" from these concepts. In addition to simplifying the schema design process, doing this also allows for these generic constructs to be used to describe devices, functions, channels, attributes, or concepts that don't fit into the more specific descriptions. These constructs are available for defining aspects of instrumentation that are vendor-specific or uncommon. The IHAL language has constructs for describing generic instruments, cards, functions, and attributes.

2.3 Utilization of Third-Party Standards

In order to avoid "reinventing the wheel," IHAL makes use of existing and emerging third-party standards for certain concepts. Portions of the integrated Network Enhanced Telemetry (iNET) program's Metadata Description Language (MDL) were imported into IHAL for defining units, buses, and measurements. Portions of the Telemetry Attributes Transfer Standard (TMATS) XML schema were imported for defining PCM data streams. Finally, the eXtensible Instrumentation Definition Markup Language (XidML) was used for defining how non-iNET network data was encoded.

2.4 IHAL Schema Overview

In this section we will provide an overview of the IHAL schema. This handbook is not meant to be a comprehensive description of IHAL, but provide enough information for a user of the IHAL schema to get started. The top-level IHAL schema diagram is shown in Figure 2-2. As shown in the diagram, the top-level elements describe use level, pool level, and some API-specific error constructs.

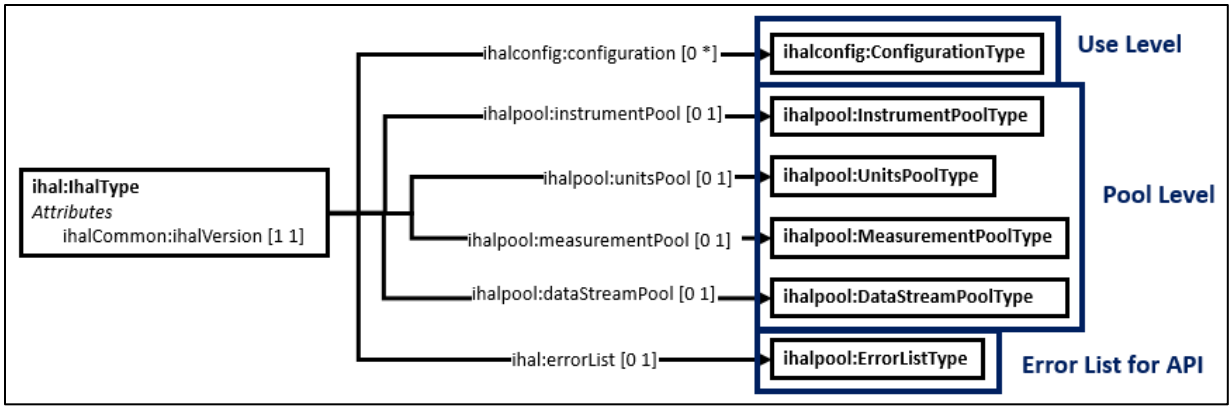


Figure 2-2. Top-Level IHAL Schema Diagram

The top-level IHAL type contains the attributes `halCommon:ihalVersion`[4], which stores the version of the schema that the instance document conforms to

The top-level IHAL type contains the following elements.

- `ihalconfig:configuration` is an element of type `ihalconfig:ConfigurationType` that describes a specific IHAL configuration (use) as described in Section 2.1. There can be 0 or more configurations in an IHAL instance document.
- `ihalpool:instrumentPool` is an element of type `ihalpool:InstrumentPoolType` that describes the devices available for including in a configuration. This is the "data sheet" information as described in Section 2.1. There can be 0 or one instrument pool in an IHAL instance document.
- `ihalpool:unitsPool` is an element of type `ihalpool:UnitsPoolType` that describes the units used in the IHAL instance document. The IHAL language uses the units defined in MDL, illustrating the use of third-party standards as defined in Section 2.3. This is an advanced topic and will not be illustrated in this handbook.
- `ihalpool:measurementPool` is an element of type `ihalpool:MeasurementPoolType` that describes the measurements used in the IHAL instance document. The IHAL language uses the measurements defined in MDL, illustrating the use of third-party standards as defined in Section 2.3. This is an advanced topic and will not be illustrated in this handbook.
- `ihalpool:dataStreamPool` is an element of type `ihalpool:DataStreamPoolType` that describes data streams used in the IHAL instance document. The IHAL language uses the data streams defined in MDL, illustrating the use of third-party standards as defined in Section 2.3. This is an advanced topic and will not be illustrated in this handbook.
- `ihal:errorList` is an element of type `ihal:ErrorListType` that describes errors returned from calls to the IHAL API. This is an advanced topic and will not be illustrated in this handbook.

[4] Recall that XML elements and types can include a namespace and an element/type name. In this case, the namespace is ihalCommon and the attribute name is ihalVersion.

The following subsections describe some of the key IHAL modeling concepts that are illustrated with specific examples.

2.4.1 **Attributes**

In IHAL, we need to be able to represent a wide range of attribute types. In this section, we describe the basic concepts for representing attributes, which in IHAL differ along two dimensions: the type of the value, and the configurability of the attribute.

The IHAL language supports numeric, string, Boolean, and reference type attributes. String and Boolean attributes have their usual meaning in IHAL. Numeric attributes are decimal XML data types, which includes reals and integers. Reference attributes are XPath[5] expressions for navigating through an XML document.

A configurable attribute is one whose value can be changed in an IHAL configuration or use. An example of a configurable attribute is the gain or offset of a signal conditioning card. A non-configurable attribute is one whose value cannot be changed in an IHAL configuration. An example of a non-configurable attribute is the connection type or weight of a signal conditioning card.

We will illustrate the IHAL attribute concept using numeric attributes. The other attribute types are easily understood once the numeric attribute concept is mastered. Figure 2-3 shows variations of the numeric type.[6] The most basic type is the `ihalattribute:NumericValueType`, which is a unitless numeric value. This is a non-configurable attribute. The `ihalattribute:NumericValueUnitType` extends the `ihalattribute:NumericValueType` type to add an MDL unit. This is a non-configurable attribute. The `ihalattribute:NumericRangeType` is a numeric type that has a minimum and maximum numeric value. This can be used in a configurable attribute since the actual value that can be assigned to the attribute in a configuration can be any value within this range. Finally, the `ihalattribute:ConfigurableNumericAttributeType` extends the `ihalattribute:NumericRangeType` to add a value step, which defines the increment between any two values within the defined range. This is a configurable attribute.

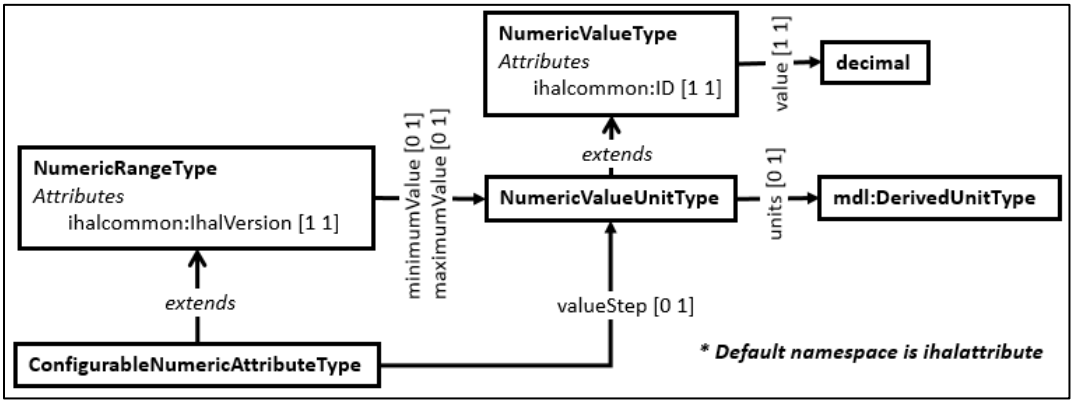


Figure 2-3. Numeric Attribute Schema Diagram

[5] "XML Path Language (XPath) Version 1.0," https://www.w3.org/TR/xpath/, Retrieved 31 January 2017.
[6] Unless otherwise noted, the namespace in this diagram is `ihalattribute`.

The IHAL "possibly configurable" attribute rolls all value types and configurability into a single concept. Figure 2-4 shows the schema for this attribute.

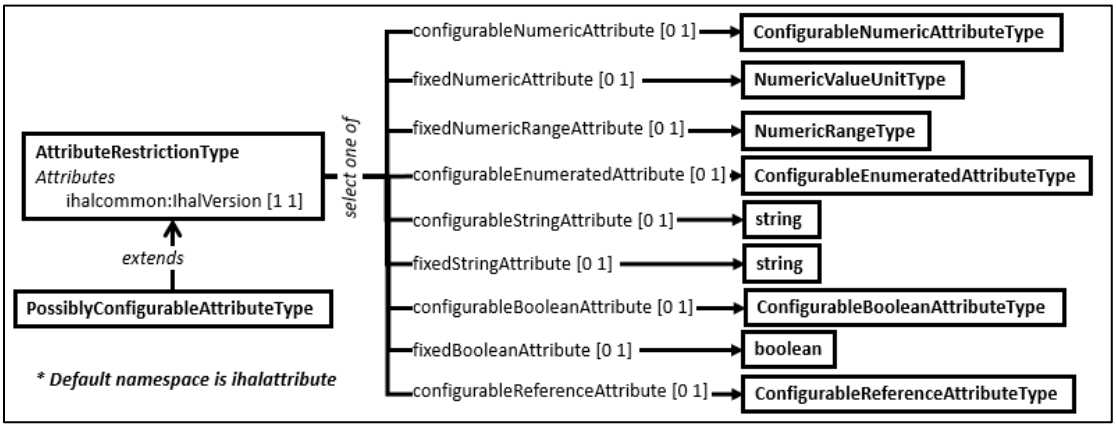


Figure 2-4. Possibly Configurable Attribute Type Schema Diagram

The `ihalattribute:PossiblyConfigurableAttributeType` type consists of *one of* the following elements:

- `ihalattribute:configurableNumericAttribute` is an element of type `ihalattribute:ConfigurableNumericAttributeType` that describes a configurable numeric attribute as described in Figure 2-3.
- `ihalattribute:fixedNumericAttribute` is an element of type `ihalattribute:NumericValueUnitType` that describes a non-configurable numeric attribute as described in Figure 2-3.
- `ihalattribute:fixedNumericRangeAttribute` is an element of type `ihalattribute:NumericRangeType` that describes a non-configurable numeric attribute with a range as described in Figure 2-3.
- `ihalattribute:configurableEnumeratedAttribute` is an element of type `ihalattribute:ConfigurableEnumeratedAttributeType` that describes a configurable attribute that takes its value from a list of possible values.
- `ihalattribute:configurableStringAttribute` is an element of type `string` that describes a configurable string-valued attribute.
- `ihalattribute:fixedStringAttribute` is an element of type `string` that describes a non-configurable string-valued attribute.
- `ihalattribute:configurableBooleanAttribute` is an element of type `ihalattribute:ConfigurableBooleanAttributeType` that describes a configurable Boolean-valued attribute.
- `ihalattribute:fixedBooleanAttribute` is an element of type `Boolean` that describes a non-configurable Boolean-valued attribute.
- `ihalattribute:configurableReferenceAttribute` is an element of type `ihalattribute:ConfigurableReferenceAttributeType` that describes a configurable reference-valued attribute.

The IHAL schema contains attribute definitions for some of the most common attribute types (gain, offset, excitation voltage, etc.). In cases where the IHAL user needs to define their

own attributes, we provide the `ihalattribute:CustomAttributeType` type, as shown in Figure 2-5. This type extends the `ihalattribute:PossiblyConfigurableAttributeType` defined above and adds an element to provide a description of the attribute.

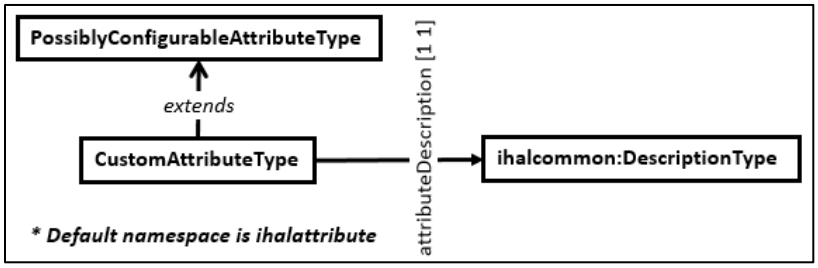


Figure 2-5. Custom Attribute Schema Diagram

2.4.2 **Generic Instruments**

We conclude the overview of the IHAL schema with a discussion of generic instruments. The IHAL language generic instruments serve two purposes: (1) they define common attributes and elements that more specific instrument types in the IHAL schema (DAUs, signal conditioning cards, encoders, etc.) inherit from; and (2) they provide a way to add new types of instruments that are not included in the IHAL schema.

In IHAL, there is a more generic concept than the generic instrument, and that is the generic IHAL device. The generic device describes IHAL "things" using the most common elements. Figure 2-6 shows the generic IHAL device[7], and contains the following elements:

- `ihaldevice:deviceDescription` is an element of type `ihaldevice:DeviceDescriptionType` that contains descriptive information (device name, device description, a custom identifier, manufacturer name, part number, and one or more additional descriptions) about the device. Details of this type are shown in Figure 2-7.
- `ihaldevice:dimensions` is an element of type `ihaldevice:DimensionsType` that contains size information (device shape, device location, device length, device width, and device height) about the device. Details of this type are shown in Figure 2-8.
- `ihaldevice:baseWeight` is an element of type `ihalattribute:NumericValueUnitType` that contains the weight of the device. Details of this type were described above.
- `ihaldevice:connector` is an element of type `ihaldevice:ConnectorType` that contains a description of the device's connectors (connector medium, port direction, plug/socket type, connector standard). Details of this type are shown in Figure 2-9.
- `ihaldevice:operationalEnvironmentCharacteristics` and `ihaldevice:nonOperationalEnvironmentCharacteristics` are elements of type `ihalenvironmental:EnvironmentalCharacteristicsType` that contains information about the environmental operational and non-operational conditions (packaging, temperature ranges, vibration bound, acceleration bound, humidity ranges,

[7] Unless otherwise noted, the namespace in this diagram is `ihaldevice`.

altitude ranges, shock ranges, acoustic noise bounds, and electromagnetic interference ranges). Details of this type are shown in Figure 2-10.

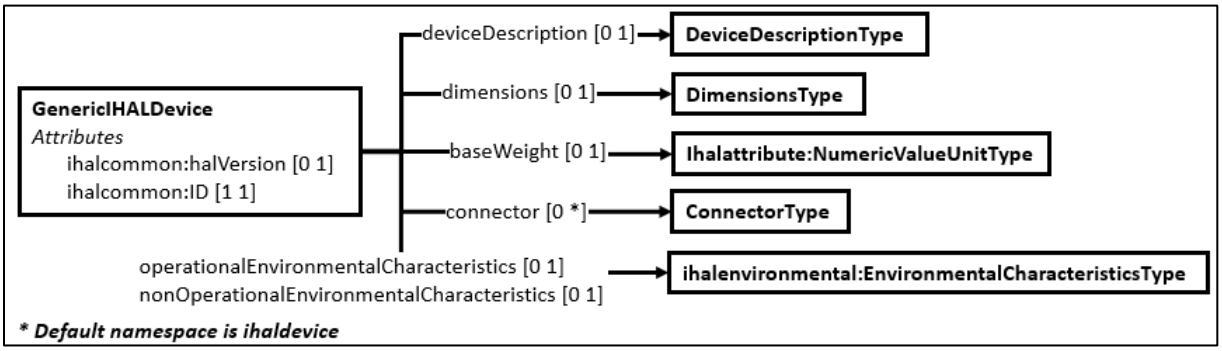


Figure 2-6. Generic IHAL Device Schema Diagram

Figure 2-7 shows the device description schema diagram.

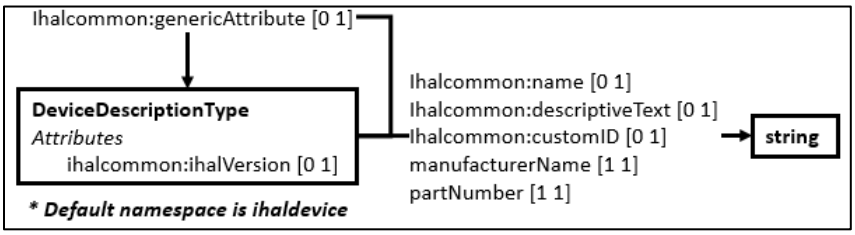


Figure 2-7. Device Description Schema Diagram

Figure 2-8 shows the dimensions schema diagram.

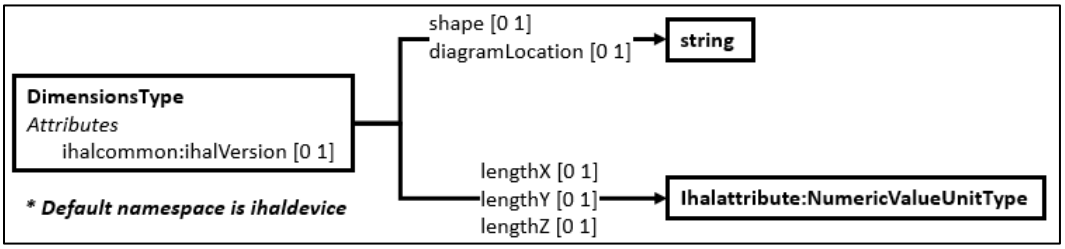


Figure 2-8. Dimensions Schema Diagram

Figure 2-9 shows the connector schema diagram.

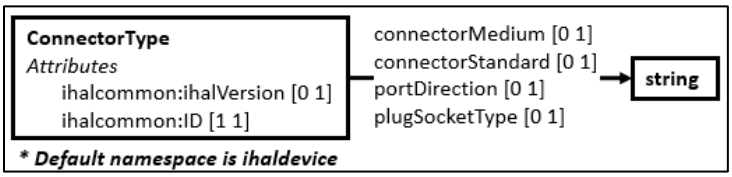


Figure 2-9. Connector Schema Diagram

Figure 2-10 shows the environmental schema diagram. The details of each of the environmental characteristics are outside the scope of this handbook. Refer to the IHAL schema for details.

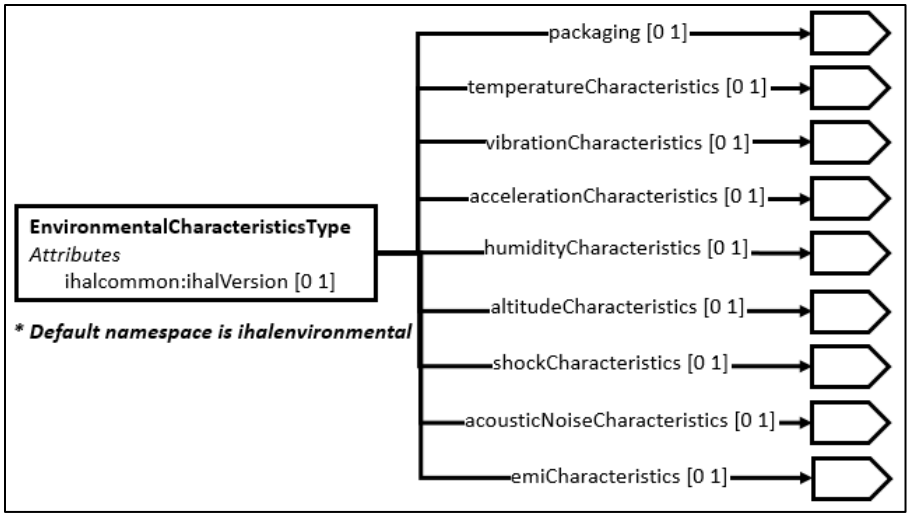


Figure 2-10. Environmental Characteristics Schema Diagram

Now that we have defined the generic IHAL device, we can define the generic IHAL instrument. It is at this point that we will also introduce IHAL functions and channels (described below). A generic instrument in IHAL extends the generic IHAL device and consists of a list of functions and a list of channels. Figure 2-11 shows the generic IHAL instrument[8], and contains the following elements.

- `ihalinstrument:additionalHardwareFunctions` is an element of type `ihalfunction:HardwareFunctionsListType` that contains the list of functions for the device. The `ihalfunction:HardwareFunctionsListType` type, in turn, contains the following elements.
  - `ihalfunction:customFunction` is an element of type `ihalfunction:GenericFunctionType` that allows the user to define a custom function (a function that is not included in the IHAL schema). The structure of the `ihalfunction:GenericFunctionType` is described below.
  - There are additional elements for specific function types (signal conditioning, amplification, analog-to-digital conversion, filtering, etc.) represented by the element "specific function" in this diagram[9]. The details of each of these functions are outside the scope of this handbook. Refer to the IHAL schema for details. Some of these functions will be described in the following chapter in the context of illustrative examples.
- `ihalinstrument:additionalHardwareChannels` is an element of type `ihalinstrument:HardwareChannelListType` that contains the list of channels for the device. The `ihalinstrument:HardwareChannelListType` type, in turn, contains the following elements.
  - `ihalinstrument:customHardwareChannel` is an element of type `ihalinstrument:GenericHardwareChannelType` that allows the user to define

[8] Unless otherwise noted, the namespace in this diagram is `ihalinstrument`.
[9] In the IHAL schema, there is not an element named "specific function." This is just a shorthand that represents all the possible functions that can be attached to the generic instrument.

a custom channel (a channel that is not included in the IHAL schema). The structure of the `ihalinstrument:GenericHardwareChannelType` is described below.

- There are additional elements for specific channel types (signal conditioning channel, bus monitor channel, etc.) represented by the element "specific channel" in this diagram.[10] The details of each of these channels are outside the scope of this handbook. Refer to the IHAL schema for details. Some of these channels will be described in the following chapter in the context of illustrative examples.

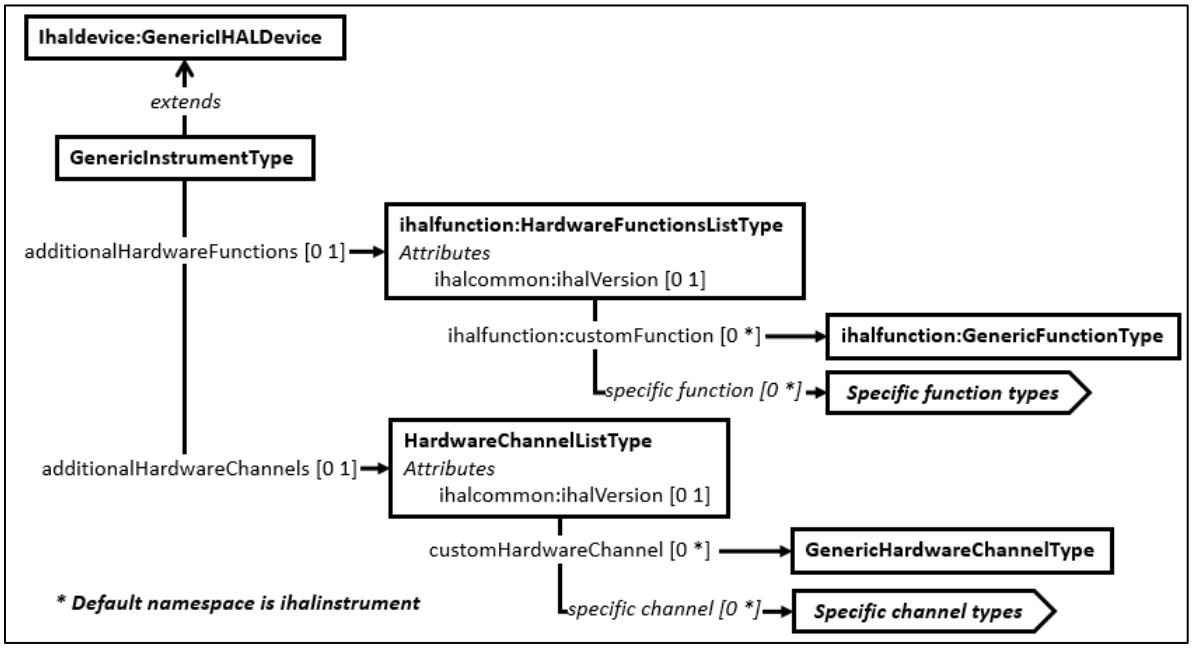


Figure 2-11. Generic Instrument Schema Diagram

2.4.3 Generic Functions

In IHAL, the functionality of each device is captured by defining one or more functions that the device can perform. Each of these functions can have associated settings, which we call attributes or more specifically, possibly configurable attributes in IHAL (described earlier). Additionally, some functions can be decomposed into sub-functions. This section describes the IHAL generic function. The next chapter will describe specific functions in the context of an illustrative example. Figure 2-12 shows the generic IHAL function[11], and contains the following elements.

- `ihalfunction:functionDescription` is an element of type `ihalcommon:DescriptionType` (see Figure 2-7) that contains a description of the function.
- `ihalfunction:customFunctionCharacteristics` is an element of type `ihalfunction:GenericFunctionCharacteristicsType` that contains the details

[10] In the IHAL schema, there is not an element named "specific channel." This is just a shorthand that represents all the possible channels that can be attached to the generic instrument.
[11] Unless otherwise noted, the namespace in this diagram is `ihalfunction`.

of the function. The `ihalfunction:GenericFunctionCharacteristicsType` type, in turn, contains the following elements.

- `ihalfunction:additionalSubFunctions` is an element of type `ihalfunction:HardwareFunctionsListType` that contains possible sub-functions for the function. The `ihalfunction:HardwareFunctionsListType` was described in Figure 2-11.
- `ihalfunction:functionEnabled` is an element of type `ihalattribute:PossiblyConfigurableBooleanAttributeType` that indicates if the function is enabled or not. This is a configurable attribute that can be set for a specific IHAL configuration.
- `ihalfunction:customAttribute` is an element of type `ihalattribute:CustomAttributeType` that allows the user to define attributes that are specific for this function.

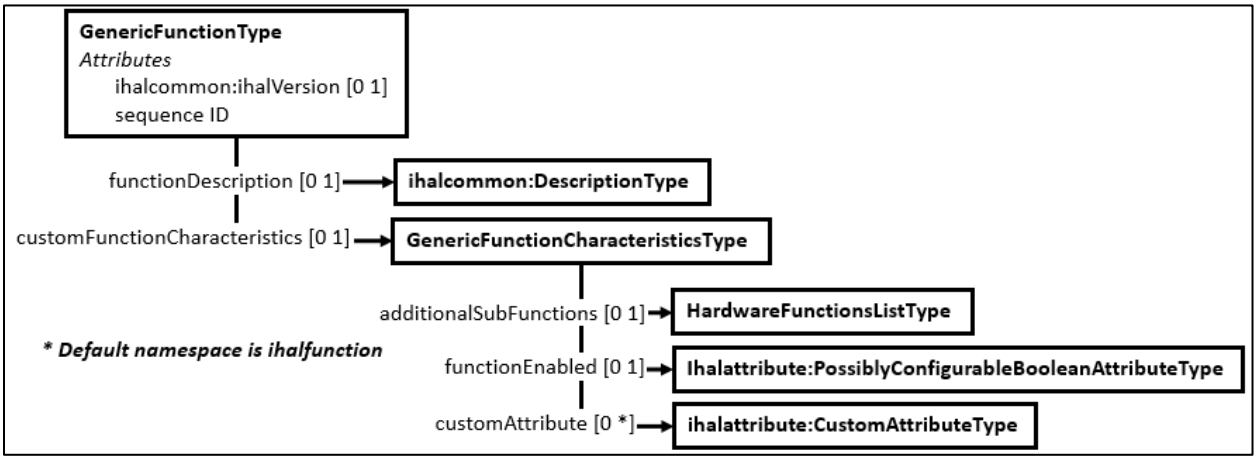


Figure 2-12. Generic Function Schema Diagram

2.4.4 Generic Channels

In IHAL, a channel is defined by the functions that the channel can perform; a multiplicity (the number of channels that have identical descriptions); and a reference to the connector(s) that the channel is associated with. This section describes the IHAL generic channel. The next chapter will describe specific channels in the context of an illustrative example. The generic IHAL channel extends an IHAL concept called `ihalfunction:GenericFunctionalityGroupingType` that represents a grouping of functionality, possibly with a multiplicity. For example, a channel might perform an amplification function and the device might have four such channels; in this case, the multiplicity would be four. Figure 2-13 shows the generic IHAL channel[12], and contains the following elements.

- `ihalinstrument:additionalHardwareFunctions` is an element of type `ihalfunction:HardwareFunctionsListType` that contains possible sub-functions for the function. The `ihalfunction:HardwareFunctionsListType` was described in Figure 2-11.

[12] Unless otherwise noted, the namespace in this diagram is `ihalinstrument`.

- `ihalinstrument:connectorRefs` is an element of type `ihalattribute:PossiblyConfigurableBooleanAttributeType` that indicates if the function is enabled or not. This is a configurable attribute that can be set for a specific IHAL configuration.

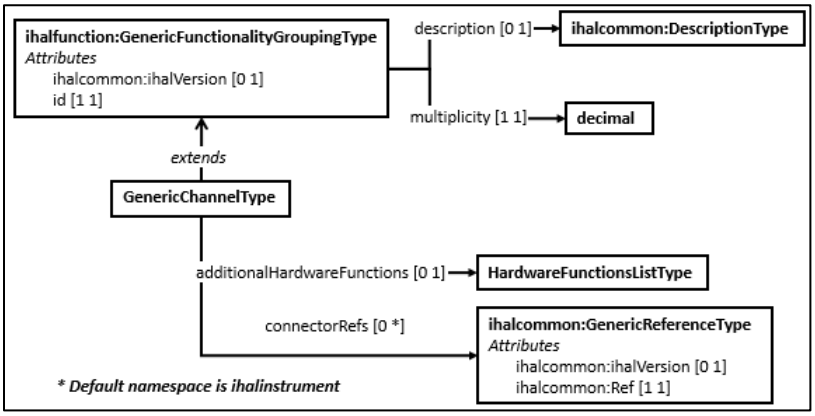


Figure 2-13. Generic Channel Schema Diagram

This page intentionally left blank.

CHAPTER 3

Getting Started with a Simple Example

3.1 Basic Instrumentation Graph

Let's start with an example consisting of a simple configuration with an empty instrumentation graph. An instrumentation graph in IHAL represents a collection of interconnected instrumentation devices. A configuration can contain more than one instrumentation graph. In this simple example, we are defining an instrumentation graph schema for an F-16 to be flown/tested at Edwards Air Force Base. The configuration and instrumentation graph are described by a unique identifier, name, and description. Table 3-1 shows the information for our configuration and instrumentation graph.

Table 3-1. Configuration/Instrumentation Graph Datasheet Information

Property	Value
Configuration name	Handbook simple example
Configuration ID	gonfig1
Configuration description	A simple example for the IHAL handbook
Instrumentation graph name	Instrumentation graph for handbook example
Instrumentation graph ID	graph1

Figure 3-1 shows the partial XML schema for this example.[13] Figure 3-2 shows the XML snippet for this example.

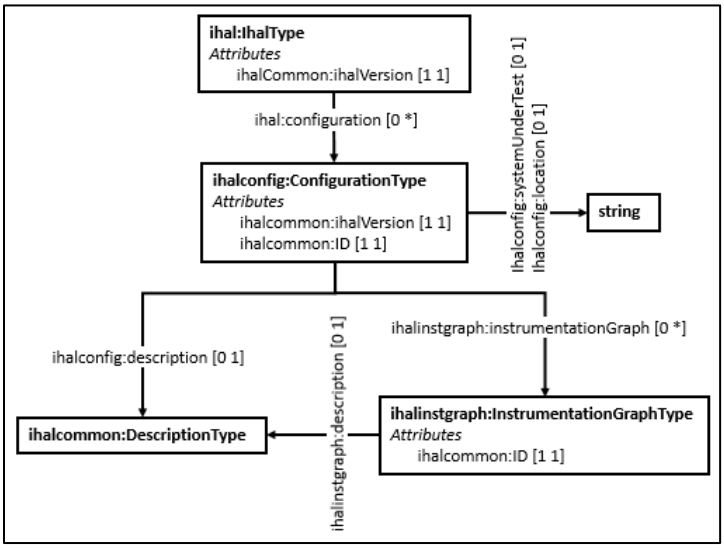
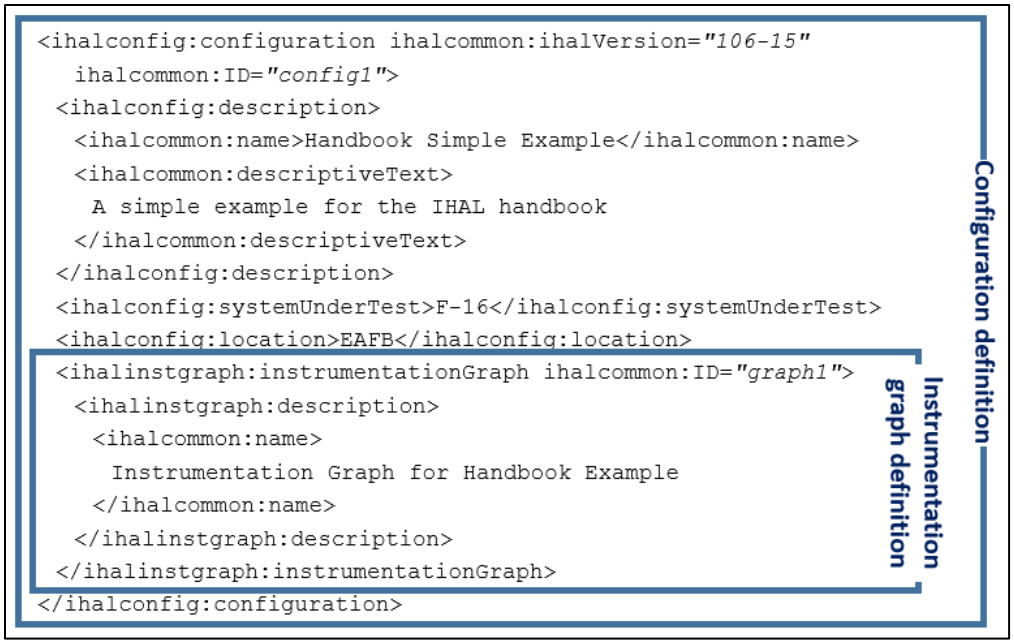


Figure 3-1. A Simple IHAL Example XML Schema

[13] In contrast to the previous section, we will only illustrate the portions of the respective schemas that are relevant to the examples presented in this section, and only describe portions of the schema that are of particular note. Details can be found in the published schema.




```
<ihalconfig:configuration ihalcommon:ihalVersion="106-15"
   ihalcommon:ID="config1">
  <ihalconfig:description>
   <ihalcommon:name>Handbook Simple Example</ihalcommon:name>
   <ihalcommon:descriptiveText>
    A simple example for the IHAL handbook
   </ihalcommon:descriptiveText>
  </ihalconfig:description>
  <ihalconfig:systemUnderTest>F-16</ihalconfig:systemUnderTest>
  <ihalconfig:location>EAFB</ihalconfig:location>
  <ihalinstgraph:instrumentationGraph ihalcommon:ID="graph1">
   <ihalinstgraph:description>
    <ihalcommon:name>
     Instrumentation Graph for Handbook Example
    </ihalcommon:name>
   </ihalinstgraph:description>
  </ihalinstgraph:instrumentationGraph>
</ihalconfig:configuration>
```

Figure 3-2. A Simple IHAL Example.

In the following sections, new details will be added to the simple example to create a complete IHAL description of an instrumentation system consisting of a DAU with an analog signal conditioning card.

3.2 Data Acquisition Unit

Before we can add a device to our simple configuration, we must first provide a pool-level specification of the device. Remember that in the pool, a device is described according to its capabilities and configurable attributes. In this next example, we add the `ihalpool:instrumentPool` element as a new child under the top-level IHAL element and describe a simple DAU. Figure 3-3 shows the partial schema for describing DAUs in the instrument pool. We show only the master controller function in this example. We also are not showing the details of the master controller function attributes; they are described using the IHAL attributes schema described in Subsection 2.4.1.

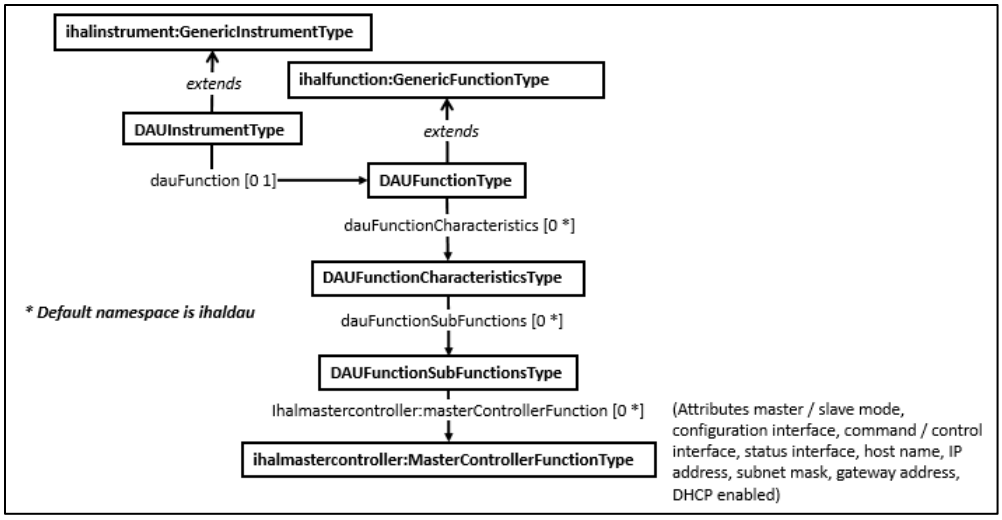


Figure 3-3. Partial DAU Schema

Table 3-2 shows the datasheet information for the DAU.

Table 3-2. DAU Datasheet Information[14]

Property	Value
DAU ID	dau1
Manufacturer	ACME
Part number	ACME-DAU-01
X dimension	10
Y dimension	20
Z dimension	10
Weight	5

Figure 3-4 shows the IHAL XML snippet with the above descriptive information for the DAU.

[14] The units in this example are not shown for simplicity reasons.

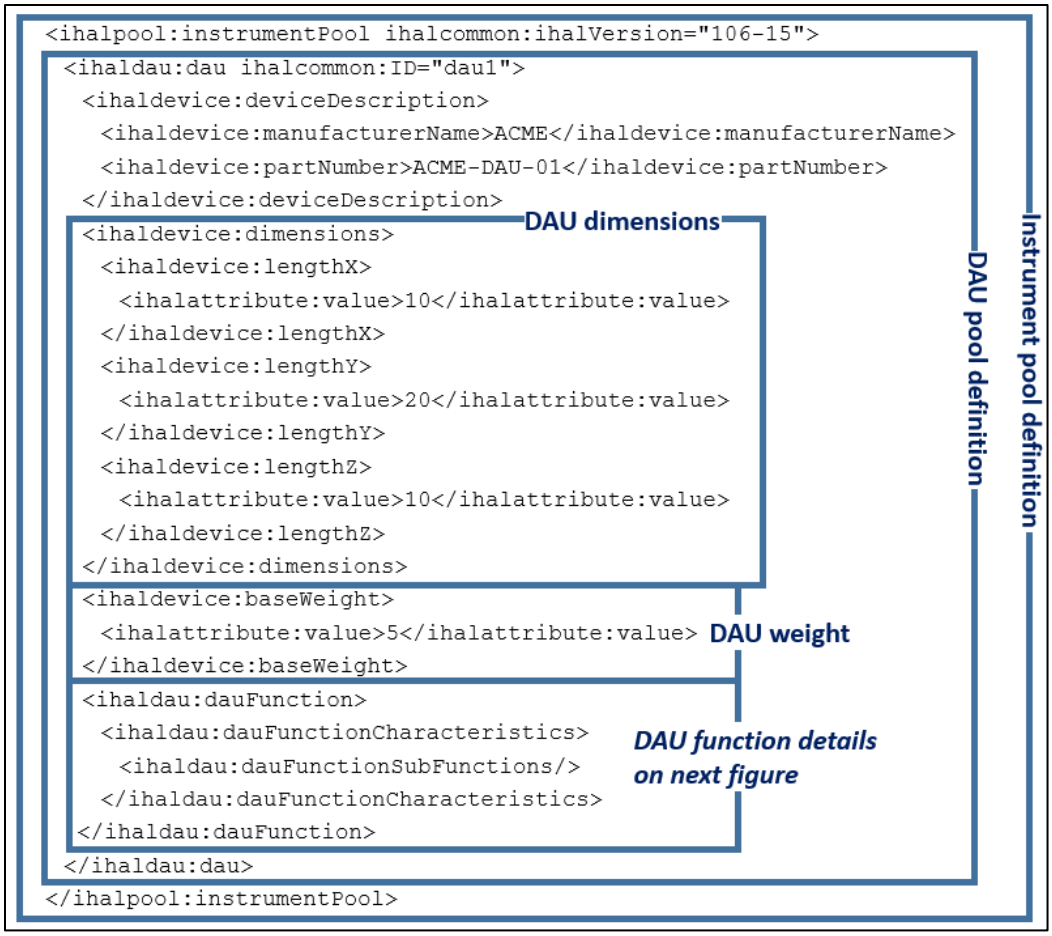


Figure 3-4. Instrument Pool with a single DAU.

In addition, the DAU can perform a master, slave, or controller role in a configuration (this is a configurable attribute). Figure 3-5 shows the IHAL XML snippet with the master controller functional description.

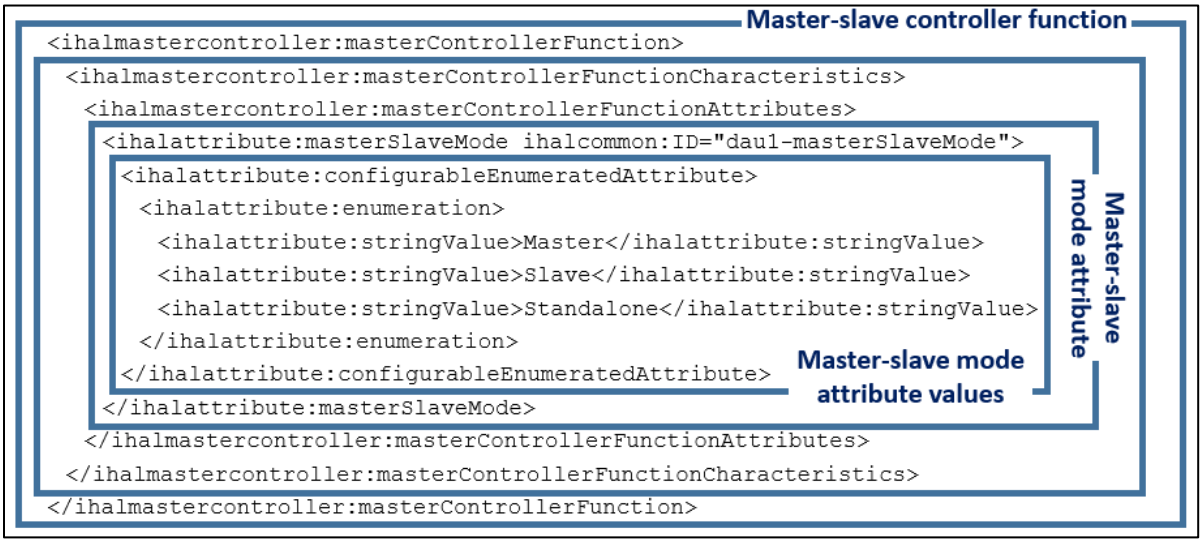


Figure 3-5. Instrument Pool DAU Function Characteristics

Now that we have a DAU described in our pool, we can use that DAU in an instrumentation network. Let's add an instance of our DAU to our instrumentation graph, specify its location as behind the cockpit, and set its master/slave mode to "Standalone" (a configurable attribute defined in the pool entry in Figure 3-5). Table 3-3 shows the details of the DAU.

Table 3-3. DAU Use Information

Property	Value
Use ID	dauUse1
ID (pool)	dau1
Serial number	A0000001
Location	Behind cockpit
Master-slave mode	Standalone

Figure 3-6 shows the partial IHAL XML schema for setting attributes. The `ihalinstuse:InstrumentationUseType` is used to ref to a specific item defined in the pool. The `ihalcommon:Ref` attribute points to the corresponding `ihalcommon:ID` of the instrument in the pool. Each instrument use can have attribute settings. The `ihalinstuse:setAttribute` element is used to set the value of a specific attribute. The `ihalcommon:Ref` attribute of the `ihalinstuse:SetAttributeType` concept points to the corresponding `ihalcommon:ID` of the attribute to set. In this case, we are setting the value of a configurable enumerated attribute type.

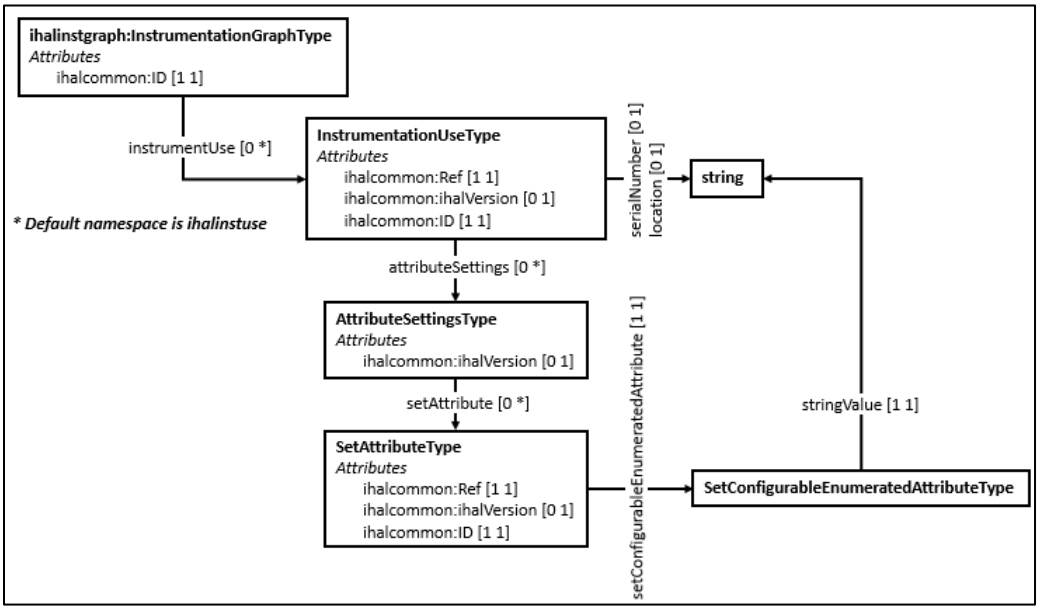


Figure 3-6. Instrument Use Set Attribute Schema

Figure 3-7 shows the XML snippet for setting the master-slave value of the DAU to "Standalone." The reference to the DAU defined in Figure 3-4 is specified using the `ihalcommon:Ref` attribute of the `ihalinstuse:instrumentUse` element. The reference to the master-slave value defined in Figure 3-5 is specified using the `ihalcommon:Ref` attribute of the `ihalinstuse:setAttribute` element.


```
<ihalconfig:configuration ihalcommon:ihalVersion="106-15"
 ihalcommon:ID="config1">
 <ihalinstgraph:instrumentationGraph ihalcommon:ID="graph1">
  <ihalinstuse:instrumentUse ihalcommon:ID="dauUse1" ihalcommon:Ref="dau1">
   <ihalinstuse:serialNumber>A0000001</ihalinstuse:serialNumber>
   <ihalinstuse:location>Behind cockpit</ihalinstuse:location>
   <ihalinstuse:attributeSettings>
    <ihalinstuse:setAttribute
     ihalcommon:ID="setMasterSlaveMode1"
     ihalcommon:Ref="dau1-masterSlaveMode">
     <ihalinstuse:setConfigurableEnumeratedAttribute>
      <ihalinstuse:stringValue>Standalone</ihalinstuse:stringValue>
     </ihalinstuse:setConfigurableEnumeratedAttribute>
    </ihalinstuse:setAttribute>
   </ihalinstuse:attributeSettings>
  </ihalinstuse:instrumentUse>
 </ihalinstgraph:instrumentationGraph>
</ihalconfig:configuration>
```

Figure 3-7. Example Instrumentation Graph with DAU Added

3.3 Signal Conditioning

Now that we have a standalone DAU, let's describe an analog signal conditioning card in the pool so we can add it to our configuration. Figure 3-8 shows the analog signal conditioning card IHAL schema. This card extends the `ihalinstrument:GenericInstrumentType` (Figure 2-11) with specific channels (see Figure 2-13 for the more general type) that implement specific analog signal conditioning functions (see Figure 2-12 for the more general type).

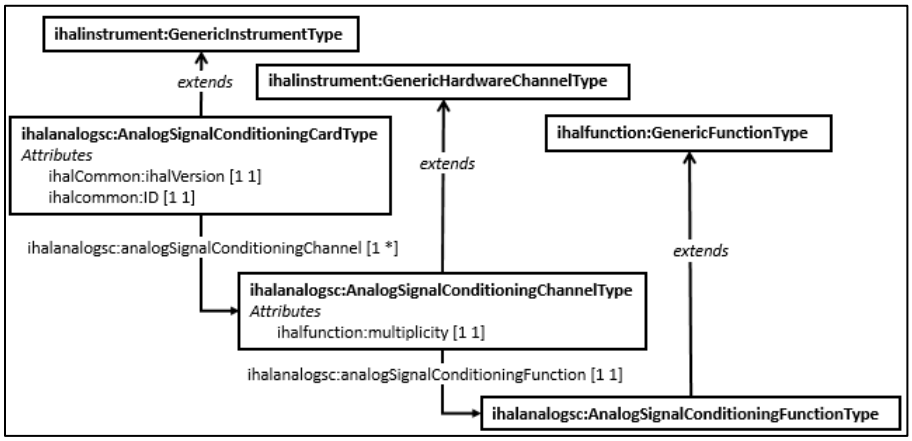


Figure 3-8. Analog Signal Conditioning Card XML Schema

Table 3-4 shows the datasheet information for the signal conditioning card. Figure 3-9 shows the IHAL analog signal conditioning function XML schema. Figure 3-10 shows the IHAL analog signal conditioning function attributes XML schema. Figure 3-11 shows the IHAL analog signal conditioning function sub-functions XML schema.

Table 3-4. Signal Conditioning Card Datasheet Information	
Property	**Value**
Manufacturer	ACME
Part number	ACME-ASC-01
Number of channels	24
Minimum input voltage	−10 to 10 V
Maximum input voltage	−10 to 10 V
Cutoff frequency	0.25, 0.50, 1.00, 2.00, 4.00, 8.00, or 16.00

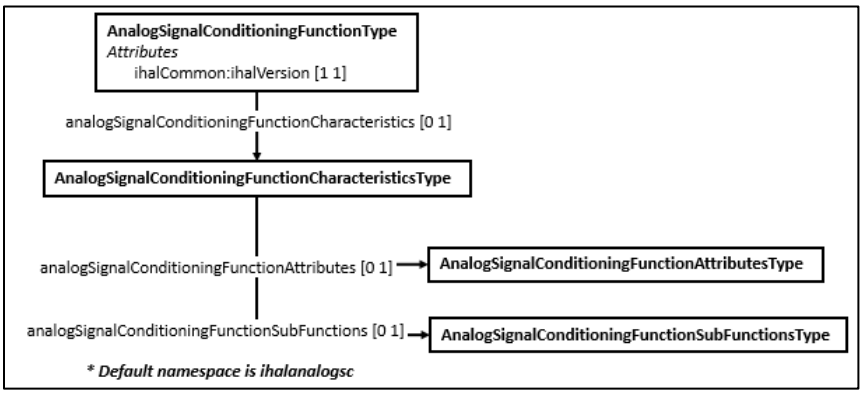


Figure 3-9. Analog Signal Conditioning Function XML Schema

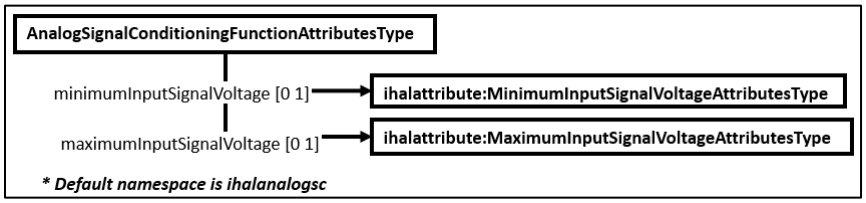


Figure 3-10. Analog Signal Conditioning Function Attributes XML Schema

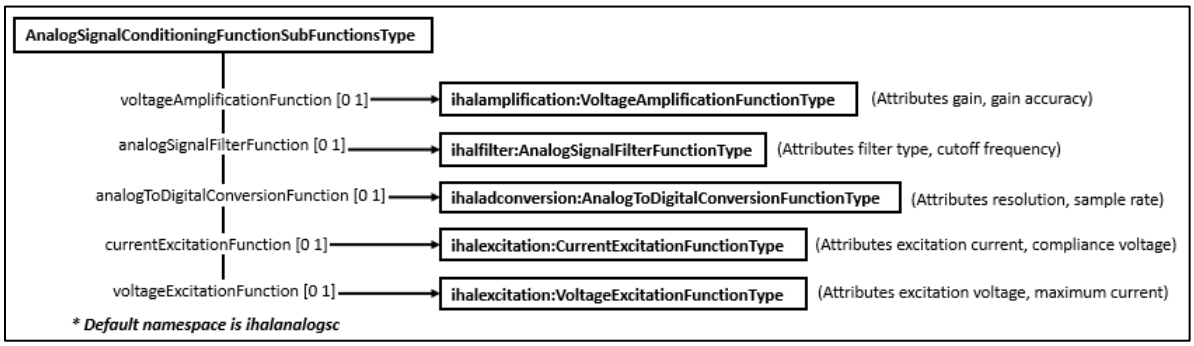


Figure 3-11. Analog Signal Conditioning Function Sub-Functions XML Schema

Figure 3-12 shows the IHAL XML snippet with the above descriptive information for the analog signal conditioning card (includes channel definition without functional details).

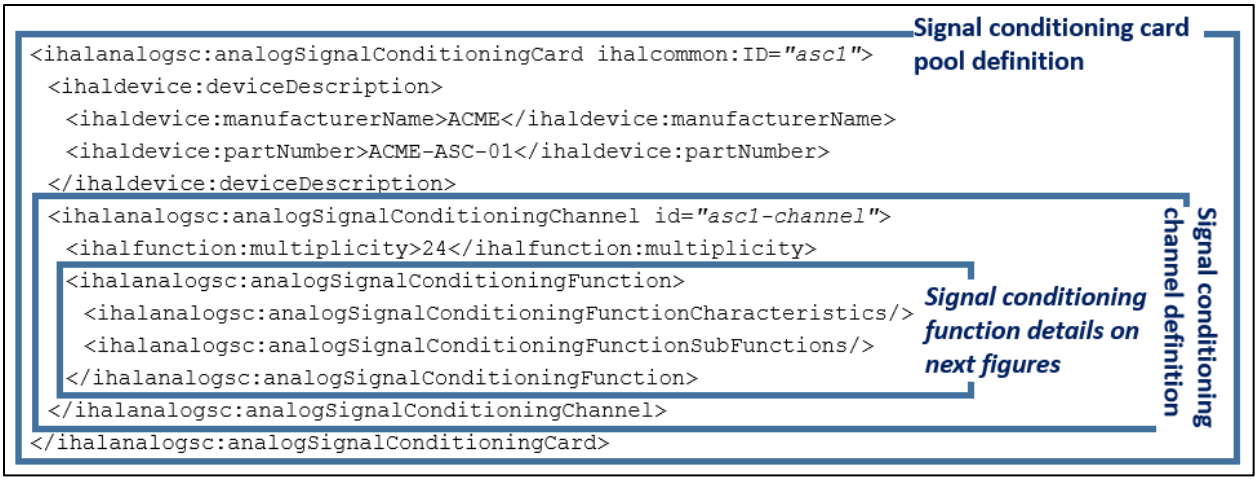


Figure 3-12. XML Snippet with Analog Signal Conditioning Card Descriptive Information

Figure 3-14 and Figure 3-14 show the IHAL XML snippet with the functional information (functional attribute minimum signal voltage and sub-function analog filter function with cutoff frequency).

Figure 3-13. XML Snippet with Analog Signal Conditioning Card Function Information

Figure 3-14. XML Snippet with Analog Signal Conditioning Card Sub-Function Information

Now let's add the analog signal conditioning card to our configuration and set some values to the configurable attributes of the channel (minimum voltage and offset pool attributes, defined in Figure 3-13). Table 3-5 shows the details of the channel use.

Table 3-5. Signal Conditioning Card Channel Use Information

Property	Value
Use ID	cardUse1
ID (pool)	asc1
Channel use ID	cardUse1Channel1
Channel ID (pool)	asc1-channel
Channel minimum input voltage	−5
Channel maximum output voltage	5
Channel cutoff frequency	0.25

Figure 3-15 shows the partial IHAL XML schema for setting channel attributes. The `ihalinstuse:ChannelUseType` is used to reference a specific channel item defined in the pool. The `ihalcommon:Ref` attribute points to the corresponding `ihalcommon:ID` of the channel in the pool. Each channel use can have attribute settings. The `ihalinstuse:setAttribute` element is used to set the value of a specific channel attribute. The `ihalcommon:Ref` attribute of the `ihalinstuse:SetAttributeType` concept points to the corresponding `ihalcommon:ID` of the channel attribute to set. In this case, we are setting the value of a configurable numeric attribute type.

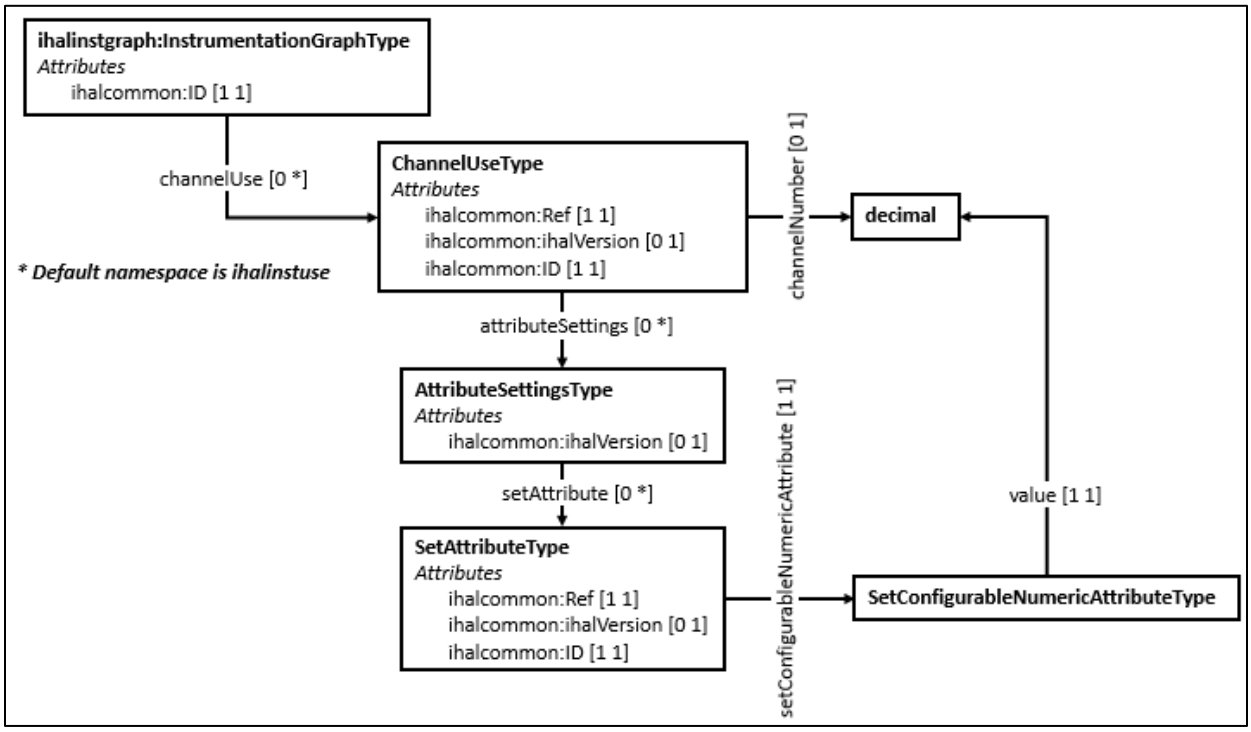


Figure 3-15. Channel Use Set Attribute Schema

Figure 3-16 shows the XML snippet for setting the minimum voltage and cutoff frequency for channel 1 of the signal conditioning card.[15] The reference to the channel defined in Figure 3-13 is specified using the `ihalcommon:Ref` attribute of the `ihalinstuse:channelUse` element. The reference to the minimum voltage and gain values defined in Figure 3-13 is specified using the `ihalcommon:Ref` attribute of the `ihalinstuse:setAttribute` element.

[15] The user can configure specific channels by referencing the channel number.

Figure 3-16. Analog signal conditioning card added to the instrumentation graph.

We now have a DAU and a card in our configuration, but they are not connected together. In reality, our card would be plugged into a slot on the DAU. In IHAL, we can add a "connection" element to our instrumentation graph to specify that two devices are connected to each other. Currently, IHAL only supports device-level or channel-level connections. A future release of IHAL will address connector- or slot-level connections.

Figure 3-17 shows the partial IHAL XML schema for defining connections. The `ihalinstgraph:connection` is used to reference a specific connection. A connection consists of two endpoints defined by a reference to the channel or slot that defines the connection.

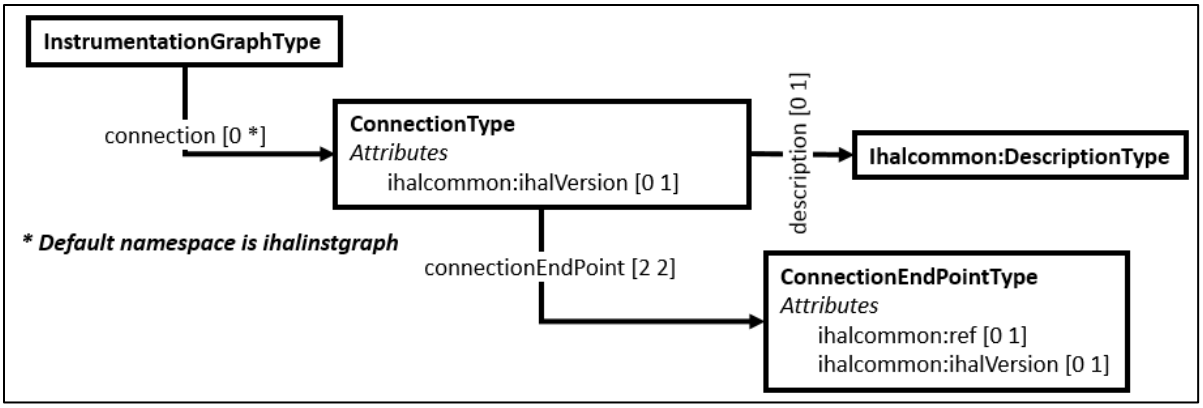


Figure 3-17. Connection Schema

The IHAL "connection" element for our example is shown in Figure 3-18.

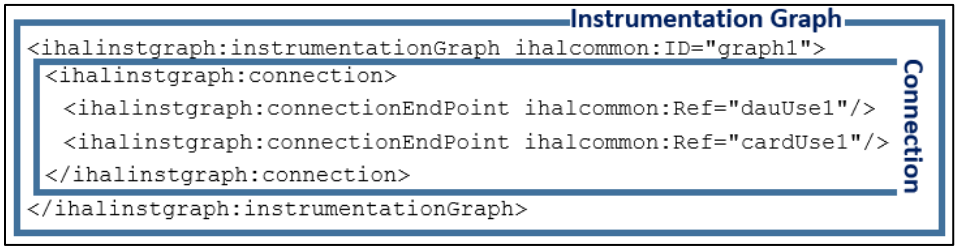


Figure 3-18. Connecting the Card to the DAU

3.4 Temperature Measurement

Following up on the illustration of modeling a signal conditioning card, we now describe how to model a sensor, specifically, a temperature measurement (thermocouple). Figure 3-19 shows the thermocouple IHAL schema. This extends the `ihalxdcr:TransducerType`, which extends the `ihalinstrument:GenericInstrumentType` (Figure 2-11).

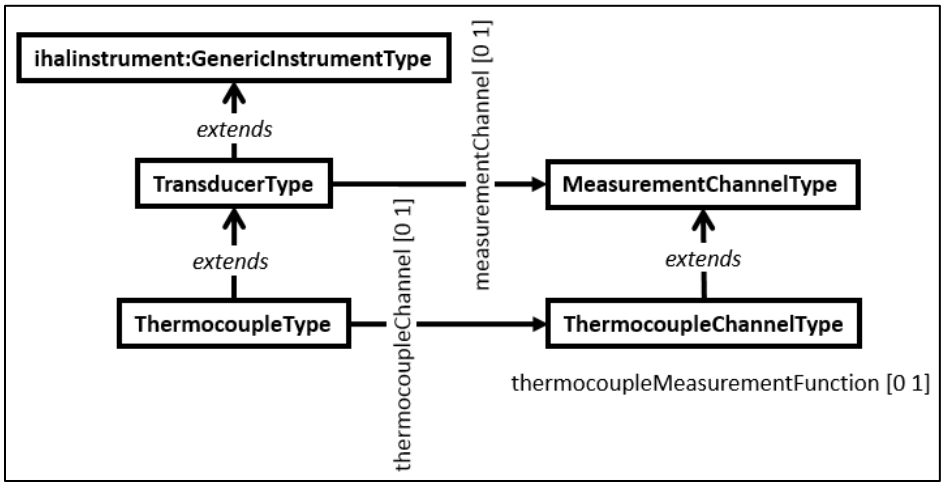


Figure 3-19. Thermocouple XML Schema

Table 3-6 shows the datasheet information for the thermocouple. Figure 3-20 shows the IHAL measurement channel XML schema. Figure 3-21 shows the IHAL measurement function characteristics XML schema. We are not showing the details of the measurement function attributes; they are described using the IHAL attributes schema described in Subsection 2.4.1. Figure 3-22 shows the IHAL thermocouple channel XML schema.

Table 3-6. Thermocouple Datasheet Information

Property	Value
Manufacturer	ACME
Part number	ACME-THERM-001
Maximum operating temperature	1172 K
Calibration type	K

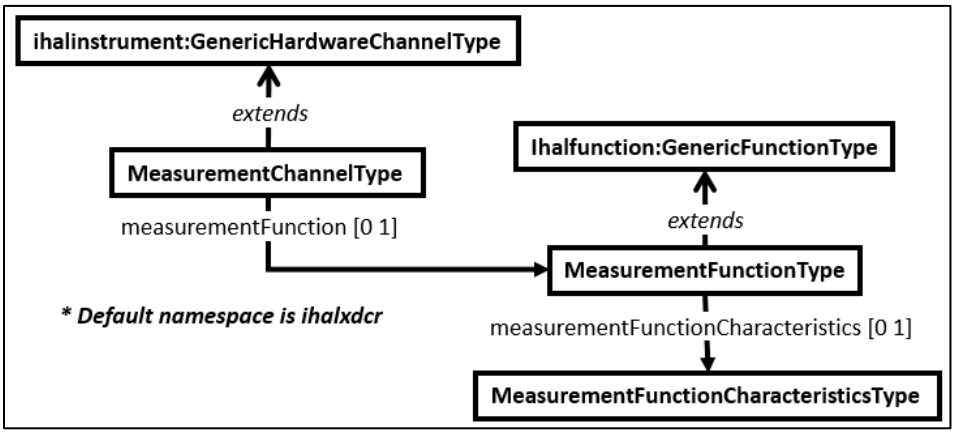


Figure 3-20. Measurement Channel XML Schema

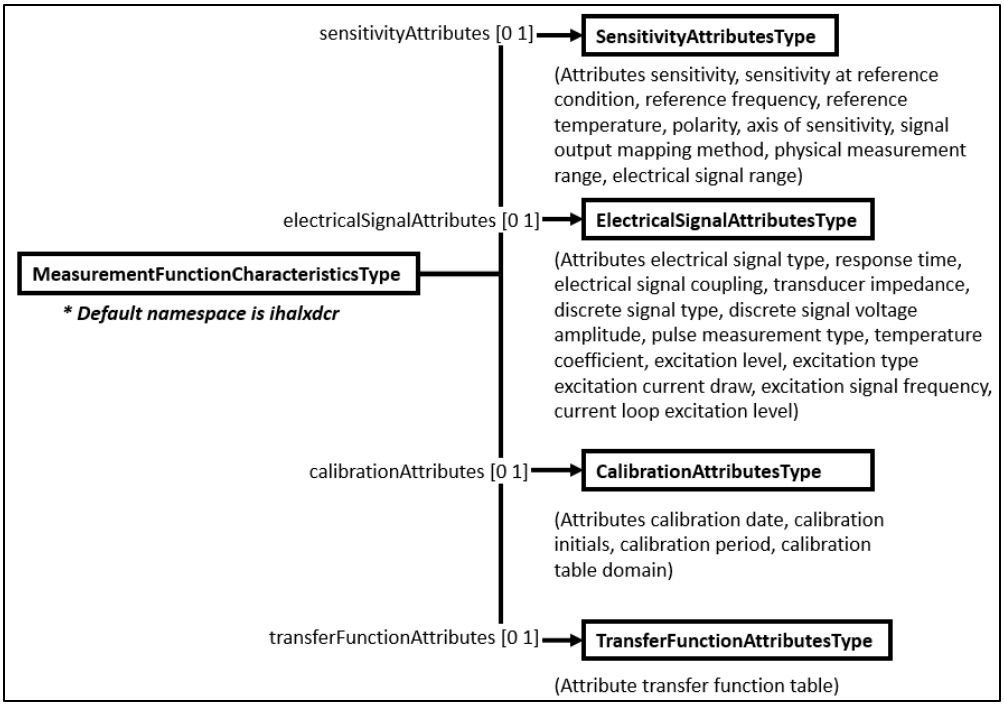


Figure 3-21. Measurement Function Characteristics XML Schema

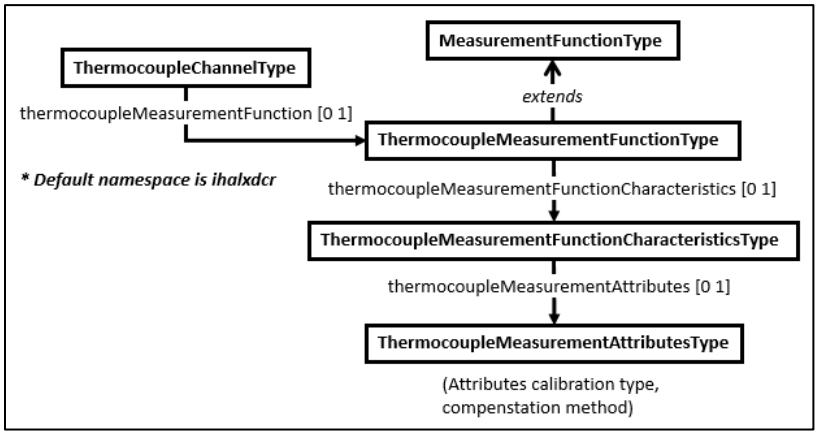
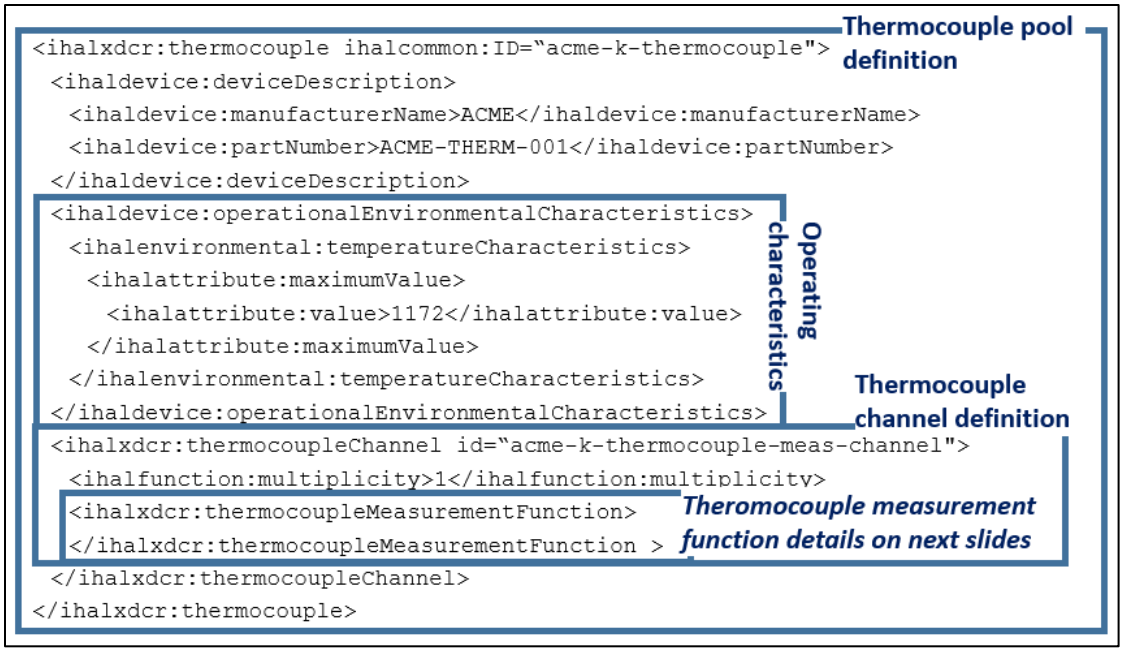


Figure 3-22. Thermocouple Channel XML Schema

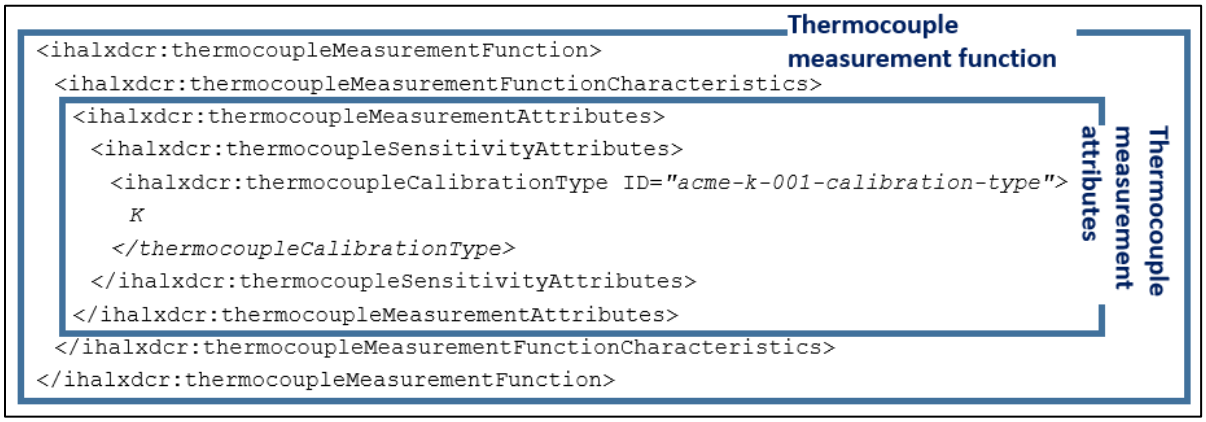
Figure 3-23 shows the IHAL XML snippet with the above information for the thermocouple (includes the channel definition without the functional details).




```
<ihalxdcr:thermocouple ihalcommon:ID="acme-k-thermocouple">
 <ihaldevice:deviceDescription>
  <ihaldevice:manufacturerName>ACME</ihaldevice:manufacturerName>
  <ihaldevice:partNumber>ACME-THERM-001</ihaldevice:partNumber>
 </ihaldevice:deviceDescription>
 <ihaldevice:operationalEnvironmentalCharacteristics>
  <ihalenvironmental:temperatureCharacteristics>
   <ihalattribute:maximumValue>
    <ihalattribute:value>1172</ihalattribute:value>
   </ihalattribute:maximumValue>
  </ihalenvironmental:temperatureCharacteristics>
 </ihaldevice:operationalEnvironmentalCharacteristics>
 <ihalxdcr:thermocoupleChannel id="acme-k-thermocouple-meas-channel">
  <ihalfunction:multiplicity>1</ihalfunction:multiplicity>
  <ihalxdcr:thermocoupleMeasurementFunction>
  </ihalxdcr:thermocoupleMeasurementFunction >
 </ihalxdcr:thermocoupleChannel>
</ihalxdcr:thermocouple>
```

Figure 3-23. XML Snippet with Thermocouple Descriptive Information

Figure 3-24 shows the IHAL XML snippet with the functional information (thermocouple calibration type).




```
<ihalxdcr:thermocoupleMeasurementFunction>
 <ihalxdcr:thermocoupleMeasurementFunctionCharacteristics>
  <ihalxdcr:thermocoupleMeasurementAttributes>
   <ihalxdcr:thermocoupleSensitivityAttributes>
    <ihalxdcr:thermocoupleCalibrationType ID="acme-k-001-calibration-type">
     K
    </thermocoupleCalibrationType>
   </ihalxdcr:thermocoupleSensitivityAttributes>
  </ihalxdcr:thermocoupleMeasurementAttributes>
 </ihalxdcr:thermocoupleMeasurementFunctionCharacteristics>
</ihalxdcr:thermocoupleMeasurementFunction>
```

Figure 3-24. XML Snippet with Thermocouple Function Information

This page intentionally left blank.

CHAPTER 4

The IHAL API

Relying only on a common language such as IHAL requires that hardware vendors expose in the IHAL format every detail of their hardware's configuration parameters that is necessary to create a valid configuration. Some of this information, such as the way configurable attributes interact with each other, is often proprietary and usually encapsulated in the vendor's configuration software (or in a third-party ISS under a non-disclosure agreement). Without this information, vendor-neutral configuration software would require the user go through the tedious process of building a configuration, loading it into the vendors' software for validation, fixing errors, and then re-loading the modified configuration until all errors are resolved.

This exposure of proprietary information can be avoided if vendors provide access to their internal validation engines through an API. Having an API allows third-party software to interact directly with the vendor's configuration software without requiring usage of the vendor's software user interface. Further, a third-party application could request individual setting changes to a configuration model in the vendor software and receive immediate feedback about the validity of these changes. If such an API were standardized so that all vendors implemented the same basic API function calls, then a third-party ISS could be developed that is capable of configuring the hardware of any vendor who implements the standard API.

When defining the specification for a standard API, interoperability is maximized if implementations of the API are not tied to a specific programming language. Such language-independence can be achieved by implementing the API as a web service, which allows disparate applications to exchange information using the common Hypertext Transfer Protocol (HTTP). Language-independence is the primary reason why this approach is preferred over the other popular approach to achieving inter-process communications: remote procedure call (RPC). Most implementations of RPC are highly dependent on the specific language or technology used to develop them, reducing interoperability by confining clients to a specific technology. To address these interoperability issues, we have designed a web-services-based API for querying and configuring vendor systems using IHAL as the communication language. The envisioned use cases for the API are shown in Figure 4-1.

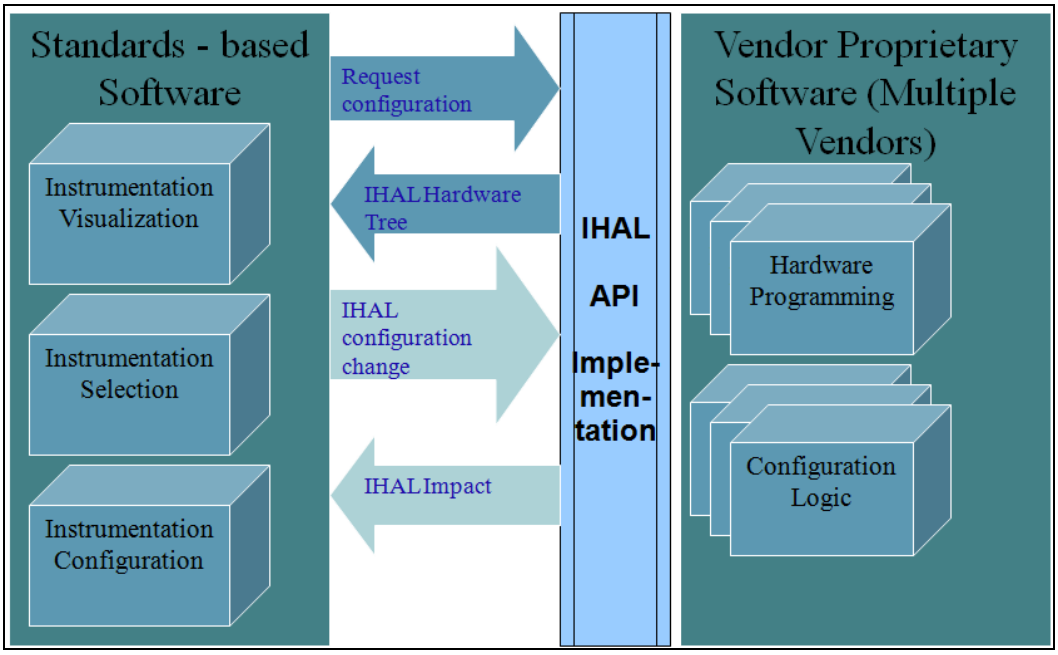


Figure 4-1. Envisioned Use Cases for the IHAL API

The IHAL configuration API was designed to be implemented as a representational state transfer (REST) web service. The decision to use REST as opposed to the Simple Object Access Protocol (SOAP) was made in order to avoid the increased complexity of SOAP, which requires that messages must be wrapped in a specifically formatted header. This additional markup makes messages larger and requires more time to transmit. In practice, the rigidity of SOAP message formats typically requires a special toolkit in the development environment to program. The main reason for this additional rigidity is to achieve strongly typed parameters. For our purposes, parameter type-checking is less important because all parameters and responses will be valid IHAL text that conforms to the IHAL XML schema.

The final API specification defines the following 11 functions:

1. Retrieve the vendor's pool-level descriptions;
2. Retrieve a list of configurations currently available in the vendor's system;
3. Retrieve the complete IHAL specification of a specific configuration;
4. Create a new configuration;
5. Modify a configuration by changing a setting;
6. Add a device to a configuration;
7. Remove a device from a configuration;
8. Program all hardware with a specific configuration;
9. Add a new format to a data encoder;
10. Add a measurement to a data encoder's format;
11. Remove a measurement from a data encoder's format.

The IHAL API must be implemented as a REST web service. All functions must have a common base path (e.g., http://10.10.1.1:8080/ihalapi/). This base path is referred to as "<vendor API Location>" in this handbook.

All inputs are provided as the payload of the function call, with no named parameters or Uniform Resource Locator (URL) encoding. That is, inputs will NOT be part of the URL (e.g., http://.../?ihal=<ihal>... is NOT allowed)

4.1 Errors

All functions in the below specification may optionally return an `<ihal:errorList>` element instead of the defined response. The error list is intended to provide the user with a description of problems encountered if the requested function could not be performed.

4.2 API Functions

The following sections describe the functions that must be included as part of any IHAL API implementation.

4.2.1 Retrieve a Vendor's Pool

This method is used by a client to retrieve some part of a vendor's pool description. There are multiple URLs for this function to retrieve different parts of the pool.

URL:	<Vendor API Location>/pool/units *to retrieve the units pool*
	<Vendor API Location>/pool/instrument *to retrieve the instrument pool*
	<Vendor API Location>/pool/measurement *to retrieve the global measurement list*
	<Vendor API Location>/pool/measurement/<deviceID> *To retrieve the list of measurements available to a particular device (e.g. a data encoder)*
	<Vendor API Location>/pool/dataStream *To retrieve the global list of data streams (e.g. buses)*
	<Vendor API Location>/pool/dataStream/<deviceID> *To retrieve the global list of data streams (e.g. buses) available to a particular device*
HTTP Verb:	GET
Function Input:	None
Return Value:	Complete IHAL <instrumentPool>, <unitsPool>, <measurementPool>, or <dataStreamPool> element.

4.2.2 Retrieve the List of Available Configurations

This function queries the web service for a list of existing instrumentation configurations.

URL:	<vendor API Location>/configurations/
HTTP Verb	GET
Function Input:	None
Return Value:	A partial <ihal> specification containing 0 or more EMPTY <configuration> elements, each with only the basic required information. No pools should be returned.

4.2.3 **Retrieve a Specific Configuration**

This function uses the ID of a configuration returned from the previous function call to request the complete description of that configuration. <configurationID> contains a unique identifier returned as the "id" attribute from a call to retrieve a list of configurations.

URL:	<vendor API Location>/configurations/<configurationID>
HTTP Verb	GET
Function Input:	None
Return Value:	A complete IHAL <configuration> element

4.2.4 **Change the Value of a Configurable Attribute**

This function is used to change the values of settings on a particular device. The desired settings changes are passed in via IHAL, and a description of everything that has changed as a result of these settings changes is returned as an IHAL description. <configurationID> contains a unique identifier returned as the "id" attribute from a call to retrieve a list of configurations.

URL:	<vendor API Location>/configurations/<configurationID>/
HTTP VERB	PUT
Function Input:	A partial <configuration> element. This element contains only the settings that the user wishes to modify.
Return Value:	The "impact": A partial IHAL <configuration> element containing only the new settings for everything that has changed: – The new values for the settings the user requested (may or may not match the original request); – Any additional settings that changed as a result; – Any attribute "restrictions" that changed as a result.

4.2.5 **Create a New Configuration**

This function is used to create a new configuration in the vendor's system. A partial or complete IHAL "configuration" element is passed as input, and the vendor responds with a validated "configuration" element that matches (as closely as possible) the input. The vendor may change use-level IDs.

URL:	<vendor API Location>/configurations/
HTTP VERB	POST
Function Input:	A partial or complete <configuration> element.
Return Value:	A validated <configuration> description that matches (as closely as possible) the input <configuration>. Use-level ID values may change.

4.2.6 **Add a Device to a Configuration**

This function is used to add a device from the pool to an existing configuration in the vendor's system. A partial or complete IHAL "instrumentUse" element is passed as input, and the vendor responds with a valid "configuration" element that includes the new device. The vendor may change use-level IDs.

URL:	<vendor API Location>/configurations/<configurationID>/devices
HTTP VERB	POST
Function Input:	A partial or complete <instrumentUse> element.
Return Value:	A valid <configuration> description that includes the new device. Use-level ID values may change.

4.2.7 **Remove a Device from a Configuration**

This function is used to remove an instrumentUse from an existing configuration in the vendor's system. The ID of the instrumentUse element is included in the URL, and the HTTP 'DELETE' verb tells the system to remove that device. The vendor must respond with a valid configuration description, with the device removed.

URL:	<vendor API Location>/ configurations/<configurationID>/devices/<instrumentUseID>
HTTP VERB	DELETE
Function Input:	None
Return Value:	A valid <configuration> description with the device removed

4.2.8 **Program the Hardware**

This function is used to tell the vendor's configuration engine to load a specific configuration onto the affected hardware. The vendor responds with a <configuration> description that includes updated values for the programming status.

URL:	<vendor API Location>/ configurations/<configurationID>/programRequest
HTTP VERB	POST
Function Input:	None
Return Value:	A partial <configuration> description with the current programming status of affected devices updated.

4.2.9 **Add a New format to a Data Encoder**

This function is used to add a new data format to a data encoder. This can be either a PCM format or an iNET network format. The client sends a partial or complete description of the format, and the vendor's service responds with an updated <configuration> element containing ONLY items that have changed (including the addition of the new format).

URL:	<vendor API Location>/ configurations/<configurationID>/<instrumentUseID>/formats
HTTP VERB	POST
Function Input:	A complete or partial format "use" description (i.e., sstFormatUse or xidMLNetworkFormatUse).
Return Value:	An updated <configuration> element containing the new format as well as any settings in the configuration that have changed as a result.

4.2.10 **Add a Measurement to an Existing Format**

This function is used to add a new measurement to an existing data format. The input uses either a XidML <Mapping> element or a TMATS <Measurement> element to describe the measurement and where it should be placed in the format. The vendor's service responds with a <configuration> element that contains a complete description of the affected format as well as any settings changes that have occurred as a result.

URL:	<vendor API Location>/ configurations/<configurationID>/<formatUseID>/measurements
HTTP VERB	POST
Function Input:	A description of the measurement and its location in the format. This will be either a XidML <Mapping> element or a TMATS-XML <Measurement> element.
Return Value:	An updated <configuration> element containing the modified format as well as any settings in the configuration that have changed as a result.

4.2.11 **Remove a Measurement from a Format**

This function is used to remove a measurement from an existing data format. The client specifies the ID of the measurement in the URL. The vendor's service must remove ALL instances of this measurement from the specified format. The service must then respond with a <configuration> element that contains a complete description of the affected format as well as any settings changes that have occurred as a result.

URL:	<vendor API Location>/ configurations/<configurationID>/<formatUseID>/<measurementID>
HTTP VERB	DELETE
Function Input:	None
Return Value:	An updated <configuration> element containing the modified format as well as any settings in the configuration that have changed as a result.

Appendix A

Citations

Range Commanders Council. "Telemetry Attributes Transfer Standard," in *Telemetry Standards*. IRIG 106-15. July 2015. May be superseded by update. Retrieved 1 July 2015. Available at http://www.wsmr.army.mil/RCCsite/Documents/106-15_Telemetry_Standards/Chapter9.pdf

———. *XML Style Guide*. RCC 125-15. July 2015. Retrieved 13 January 2017. Available at http://www.wsmr.army.mil/RCCsite/Documents/125-15_XML_Style_Guide/.

"XML Path Language (XPath) Version 1.0," https://www.w3.org/TR/xpath/, Retrieved 31 January 2017.

********** END OF IHAL HANDBOOK **********